Object Class Recognition and Localization using a Visual Vocabulary Framework

Derk Crezee

June 2007

Object Class Recognition and Localization using a Visual Vocabulary Framework

Master's thesis Artificial Intelligence

Specialization Intelligent Systems

Derk Crezee

Under supervision of:

Dr. ir. B.J.A. Kröse
A. Noulas
Z. Zivkovic

Informatics Institute, Faculty of Science

Universiteit van Amsterdam

FACULTEIT DER NATUURWETENSCHAPPEN, WISKUNDE EN INFORMATICA

June 2007

Amsterdam

Abstract

In this thesis we focus our research on automatic recognition and localization of object classes. The approach we have adopted employs the visual vocabulary framework, where a visual vocabulary is derived by clustering over a set of filter responses. Image descriptors, derived with use of a visual vocabulary, are used to generalize object classes and build class models. Several appearance based object class models are learned from a set of training images, which allow us to classify and localize objects from a number of object classes.

We have experimented with various classification techniques as well as implemented two localization algorithms. The most common approach to object localization, the tile-based approach, and a novel localization algorithm are compared. The novel technique is based on a histogram tracking approach. In contrast to the tile-based approach, where a set of image patch are evaluated to determine if an object is present, the novel approach optimizes a set of image regions, with respect to their shape, size and location. The aim is to improve the efficiency as well as the performance of localization. Additionally, this framework allows us to track and classify object simultaneously.

We show that a high classification accuracy can be obtained by using the visual vocabulary. Furthermore, the comparison between the tile-based approach and the novel approach to object localization, illustrates that the novel approach, based on histogram tracking, shows a significant improvement of efficiency and in some cases also improves localization performance.

Acknowledgments

Here I would like to thank everyone that contributed to the successful completion of this master thesis. First and foremost, I would like to thank my supervisors, Ben Kröse, Zoran Zivkovic, and Athanasios Noulas. You have all helped me to achieve a higher understanding in the world of image processing and statistical pattern recognition. In particular I am grateful for the dedicated involvement of Zoran and Athanasios, which made the weekly meetings both useful and amusing.

Furthermore, I would like to thank my dear friends Julian and Ivo. Although you did not contribute to this thesis in any specific way, the past six years with you have been stimulating. Your friendship, as well as your mutual interest in the field have supported me over the years.

Also, I would like to thank Boris for the months of companionship during the final stages of my master thesis. I hope we never have to spend another day in the Bushuis again.

Finally, there is one last person to thank. Helena, thank you for your unconditional love, friendship and devotion.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the last decade we have witnessed a dramatic increase in the number of multimedia files available on the internet. Websites such as *youtube*, *myspace* and *flickr* are flourishing and consumers are rapidly adapting themselves to this digital revolution. They are taking the opportunity to create, transmit and share an endless amount of multimedia files, resulting in enormous collections of data items of different modalities (text, audio, images, animations and video).

Largely, this increase is due to technological advances in consumer level products. Camera's and digital image recorders are becoming small and affordable. Most mobile phones you can buy in stores at the moment, have a camera integrated, capable of capturing, storing and presenting digital media. With the expansion of power of digital devices in combination with a rapid drop in cost, the number and size of multimedia collections will only increase.

With the increase of multimedia data collections, the need for effective retrieval and management of data also increases, since the volume of data is considerably more then a person can browse through in order to find desired data items. The area of research concerning this problem is the area of multimedia information retrieval. Multimedia information retrieval is based on representational scheme's of media content. In the case of textual media, content can be described using the words used in the text. However, when we are dealing with other types of media, such as images, the type of descriptors will be totally different. Content descriptors for imagery include features such as color, shape, texture, or spatial relationships of local descriptors.

The difference between human and machine interpretation of images, is that humans represent image on a conceptual level, whereas machine can only represent images on a statistical level. For this reason, the aim of image retrieval techniques is the integration of low-level visual features, i.e. statistical information such as color, shape and texture, and high-level semantic features, i.e. conceptual labels such as object information and scene categorization. The problem of bridging the gap between visual features and semantic features has been one of the biggest challenges within this field of research, also known as the *semantic gap* [1].

In this thesis we have focussed our research on object recognition and localization.

These areas of research can be seen as subfields of multimedia information retrieval and aim to derive semantical information from images. Image recognition tries to determine the type of object that is depicted inside an image and localization attempts to extract information concerning the location of certain objects within an image. This semantical information is extremely useful for efficient search and management of image collections, as well as video collections. Furthermore, this information can be used in extraction of higher-level semantical features, such as scene characteristics.

1.1. Object Recognition

Object recognition can be divided into two main research areas: object recognition and object class recognition, also referred to as generic object recognition. These are two different problems. Object recognition deals with the problem of identifying individual objects. In other words, we need to separate one object from all other objects. Thus, two objects of a similar class should be distinguished from each other. Object recognition is a hard problem, because we need to account for various image variations. Variations in viewpoint and illumination conditions, as well as scale and orientation, change the appearance of objects. Other image conditions such as occlusion and truncation decrease the viewable object surface, thus introducing noise into the object class models. Object class recognition addresses the problem of discriminating between objects of one class and those of other classes. In this case, an object class should be generalized to such a degree that two objects of the same class are labeled similarly. The problem shifts from learning a specific object model, to learning a generalized object class model. On top of the issues we face with object recognition, the added challenge is to find class models that are invariant to changes in appearance within a class, while being discriminative enough to distinguish between objects from different classes.

A wide range of approaches to object class recognition exist in literature. Some recognition approaches use the *constellation of parts* framework [2, 3, 4], where an object is modeled by a arbitrary number of parts. The aim is to identify the part, which in turn characterize an object. In addition to the object part representation, the spatial relation between parts is modeled to assist in the recognition task. However, large variations in geometrical relation of parts make classification models diffuse. What is more, the approach is computationally expensive.

Other approaches to object recognition adopt the *bag of keypoints* approach [5, 6, 7], also referred to as the visual vocabulary framework. Here, all image features are combined into one large set. So, in contrast to the former framework, all positional information is dismissed. The image features are usually a set of gradient image filters. A visual vocabulary is constructed to derive image descriptors in the form of a histogram. The visual words are constructed using k-means clustering on a sample set of image features. The image descriptors are mappings from these image features to the visual dictionary. Originally, the approach was applied in texture classification [8, 9], where it was referred to as textons[1]. However, later the same framework was adopted for object recognition with promising results [7]. Some approaches make use of an interest point detector to

[1]The terms 'codebooks', 'textons', 'keypoints' and 'visual words' have been used in literature to describe the same concept, that is to say clusters of image features. We will adopt the term 'visual words' throughout this thesis.

determine regions that are likely to contain useful image information [5, 6].

1.2. Object Localization

The second problem, object localization, also known as object detection, is related to object class recognition. It refers to the problem of deciding whether an image contains an instance of one or more object classes. Additionally, regions in which objects are present, need to be identified. In relation to object class recognition, this problem is more difficult, since more information needs to be determined. Not only do we need to discriminate between object classes, but also be able to determine which patches within an image contain objects. In addition to the issues we need to address with object recognition, such as image variations, we are faced with new issues. The most prominent of the issues concerning object localization is the fact that localization tends to be very computationally expensive. Often, localization algorithms use a classifier to evaluate several local image patches. Due to the fact that there might be millions of possible regions, differing in shape, size and position, the computational costs can be enormous. This means that a new issue arises, namely the issue of determining appropriate image patches that need to be evaluated. Then, there is the issue of combining overlapping image patches. If both patches are likely to contain an object, do we combine them into one detection, or do we decide that we have detected multiple objects?

The most common approach to object class detection is the tile-based approach [10, 11, 12, 13, 14]. Here, we slide a tile across the image and locally classify each of such patches to determine if it belongs to one of the object classes or not. The classification of image patches can be done in multiple ways. Some methods include contextual information, such as estimated geometry and camera viewpoint, to discard several image patches beforehand [10], and so to guide the detection task. Other methods have used the ada-boost algorithm to combine a set of weak learners to speedup the localization process and effectively detect objects within images [13, 10, 14].

The visual vocabulary framework has also been used for detection [7, 15]. In some cases an interest point detector is used to determine local patches which contain useful information [15]. Localization is performed by combining the framework with latent semantic analysis, in order to find topics within the images. The topics correspond to the various object categories. The keypoints corresponding to the topics, i.e. object classes, are used to determine the local image patches that belong to instances of objects. By combining the size and location of these image patches, object localization can be performed.

1.3. Project Goals

The approach to object class recognition we have adopted employs the visual vocabulary framework, which was introduced in [7]. We will use the image descriptors, derived with use of a visual vocabulary, to generalize object classes and build class models. These allow us to classify and localize objects. In order to localize objects, we have implemented two localization techniques. We have implemented the tile-based approach as well as a novel localization technique. The novel technique is based on a histogram tracking approach, were a target object model is used to guide the search for optimal image patches.

Additionally, this framework allows us to track and classify object simultaneously.

We have chosen to adopt the visual vocabulary framework because it has been widely applied for object classification and localization in the past, with remarkable results [5, 7, 6, 10, 15]. Furthermore, this approach naturally fits together with a novel localization method we will introduce in this thesis, which makes use of histogram tracking. In contrast to the tile-based approach, we do not evaluate random image patches, but optimize an image patch with respect to the region size, shape, and location.

Our goal is to experiment with the different design choices related to the visual vocabulary framework. For this purpose, several existing classification techniques will be discussed and the results of classification will be compared. Furthermore, we aim to improve on existing localization methods, with the introduction of a novel localization algorithm. We believe that localization by means of histogram tracking will improve both performance and efficiency.

1.4. Thesis Outline

In this thesis we will present several approaches to the problems of generic object recognition and localization. Although these problems and the approaches to solving them are highly related, we will discuss the methods that we applied separately. Chapter 2 introduces the visual vocabulary framework, which we will use in both object classification and localization. Object classification and some classification techniques are discussed in chapter 3. Chapter 4 introduces two approaches to object localization, which we will compare in the experiments. In chapter 5 we present the results of the classification and localization experiments. In addition, we discuss some implementation details. Finally, in chapter 6 we discuss the results and comment on the chosen approaches. Furthermore, we make some suggestions about how to improve on the results in further research.

Chapter 2

Visual Vocabulary Framework

In this chapter we will introduce the framework which we apply in our research. The visual vocabulary framework can be described in three parts. Initially, local descriptors are extracted from the images of an image collection. These descriptors are collected and used as input for a clustering algorithm. The clustering process, also known as visual vocabulary construction, is used to identify a visual vocabulary. A new image can be expressed as a histogram over this visual vocabulary: each of the images' local descriptors are assigned to the corresponding visual word using some distance metric. Finally, the histogram created from an image is used as an image descriptor in order to make inference for the content of the image.

Words from the visual vocabulary can be seen as specific microstructures defined by the prototype responses to a set of image filters. The goal is to determine an appropriate set of visual words that permit us to describe various images by its image content. The visual vocabulary can be constructed by clustering the filter responses of all images into a small set of prototypes. The process of vocabulary construction is discussed in section 2.3 in more detail.

Given that we have successfully learned a visual vocabulary, we can describe an image by the frequency of visual words from the vocabulary. Initially, each pixel of an image is transformed into a filter response vector. Thereafter, each of the response vectors is assigned to a word from our visual vocabulary. At this point, an image can be described by the distribution over the words of the visual vocabulary. The assumption is that different object classes have distinct distributions, which will allow us to successfully a new image based on its descriptor histogram.

The following sections will discuss the three parts of our framework in more detail. Section 2.1 describes the extraction of local image features and section 2.2 discusses some of the image filters that are commonly used. In section 2.3 the vocabulary construction procedure is discussed. Finally, section 2.4 describes the extraction of image descriptors from images.

2.1. Feature Extraction

The representational scheme used to describe an image is one of the most important aspects in object classification. Without meaningful image features it is hard to capture the essence of an image. Thus, it would be hard to generalize object classes. Local image features have proven to be well-suited for this purpose. Due to the locality of the extraction, these features are robust to some image variations, such as partial observability (occlusion or truncation). For example, if we would alter an image by introducing occlusion of an object, not all of the local image features would change, just the ones in locality of the occlusion. The invariance to image variations is an important aspect of image features. Since we plan to generalize object classes, we would like that an object looks the same under different scales, viewpoint, orientation, and illumination conditions. When we change the scale of an object, or when we apply any Euclidean transformation for that matter, the image features should remain the same.

We have adopted the use of gradient based image filters. These local descriptors extract gradient information of local image patches. So, the image is described by a set of image filters, called a filter bank. The filter bank is applied to each pixel, forming an N dimensional vector, where N stands for the amount of image filters. The set of response vectors for an image, extracted by convolving the filter bank on an image, will consist of a large number of unique vectors. Since the features are extracted locally, filter responses that are extracted in close proximity to one another, will not differ significantly. Filter responses throughout the image however, can differ rigorously. Thus, it is safe to assume that numerous prototype response vectors exist, of which all other filter responses are noisy variations. The prototype response vectors, the responses that are most frequent or at least reside in similar regions of the filter space, are the visual words we wish to define.

2.2. Filter Banks

As was mentioned in the previous section, we will make use of a set of image filters, commonly referred to as a filter bank. In principal we could use any combination of filters to form a filter bank, but several standard filter banks exist in literature [7, 5, 8, 9, 16, 17]. Each filter bank has different characteristics due to the use of different filters. Some of the filter banks have the property that they are invariant to specific image conditions, such as scale or rotation. When a filter bank possess such a property, it implies that the responses to the filters will, to some degree, be independent of these specific image conditions. For example, when a filter bank is rotationally invariant, the value of the filter response will not change when arbitrary rotations are applied to an image. In the following subsections, we discuss some of these filter banks[1].

[1]All of images of filter banks were generated using the software of [17], available on http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html

2.2.1. The Schmid Filter Bank

The Schmid filter bank consists of 13 separate filters [16]. All the filters from the Schmid filter bank are rotationally invariant. The set is composed of 13 isotropic filters, which are shown in figure 2.1.

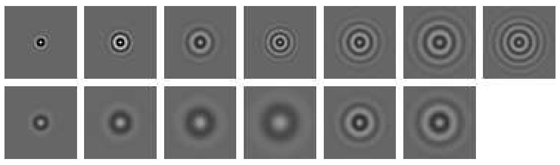


Figure 2.1.: The Schmid filter bank. As can be seen in the images, the Schmid filter bank consists of filters that are all rotationally invariant.

2.2.2. The Winn-Criminisi-Minka Filter Bank

The Winn-Criminisi-Minka filter bank consists of 11 separate filters, but a total of 17 filter responses due to convolution of the filters on different colour channels of the L,a,b colour model [7]. The filter bank is composed of 3 Gaussians ($\sigma = 1, 2, 4$), 4 Laplacian of Gaussians ($\sigma = 1, 2, 4, 8$) and 4 first order derivatives of Gaussians ($\sigma_x = 2, 4$ and $\sigma_y = 2, 4$). The 3 Gaussians are applied to all three channels of the L,a,b colour space. The 4 Laplacian of Gaussians and 4 derivatives of Gaussians are applied to the L channel only. The filters are shown in figure 2.2.

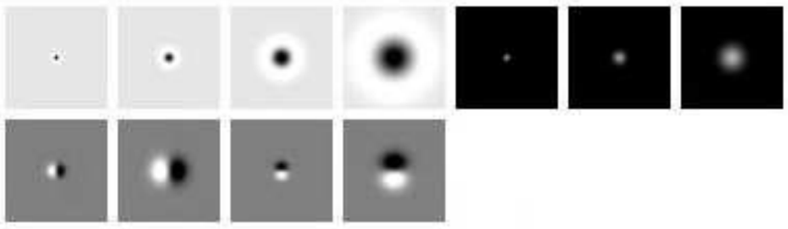


Figure 2.2.: The Winn-Criminisi-Minka filter bank. The filter bank consists of 7 isotropic filters, of which 3 Gaussians and 4 Laplacian of Gaussians, and 4 first order derivatives of Gaussians.

2.2.3. The Leung-Malik Filter Bank

The Leung-Malik filter bank consists of 48 separate filters [8, 9]. The filter bank is composed of first and second derivatives of Gaussians at 6 orientations and 3 scales. Furthermore there are 8 Laplacian of Gaussians and 4 Gaussians. The scale of the filters range from $\sigma = 1$ to $\sigma = 10$ pixels. The filters are shown in figure 2.3.

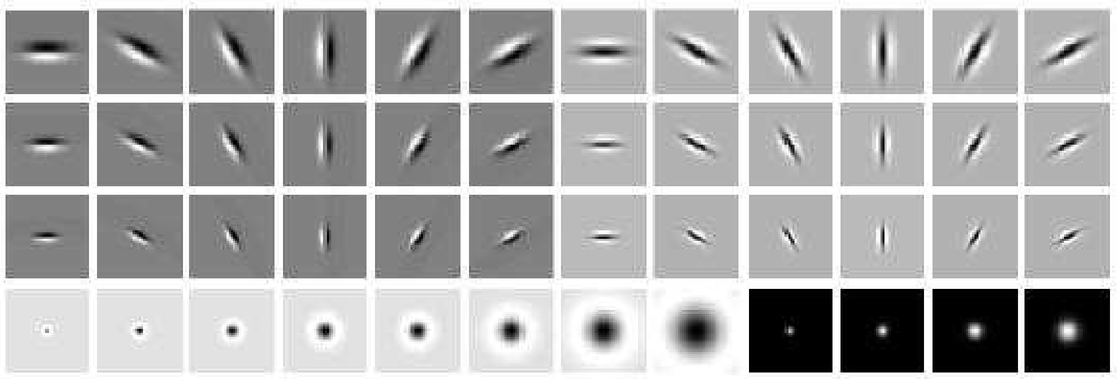


Figure 2.3.: The Leung-Malik filter bank. The filter bank consists of 48 filters, 2 anisotropic filters in 3 scale and 6 orientations, 8 Laplacian of Gaussians and 4 Gaussians.

2.2.4. The Maximum Response (MR8) Filter Bank

The MR8 filter bank consists of 38 filters, but only 8 filter responses [17]. The filter bank, like the Leung-Malik filter bank, is composed of first and second derivatives of Gaussians at six orientations and 3 scales, $(\sigma_x, \sigma_y) = \{(1, 3), (2, 6), (4, 12)\}$ pixels. The isotropic filters are a Gaussian and a Laplacian of Gaussian, both with $\sigma = 10$ pixels. The filter bank is rotationally invariant due to the fact that only 8 out of 38 filter responses are actually used. The 36 anisotropic filters are reduced to 6 by only using the maximum responses across the different orientations, making them rotationally invariant. Adding the 2 isotropic filters leaves 8 filter responses. The filters are shown in figure 2.4.

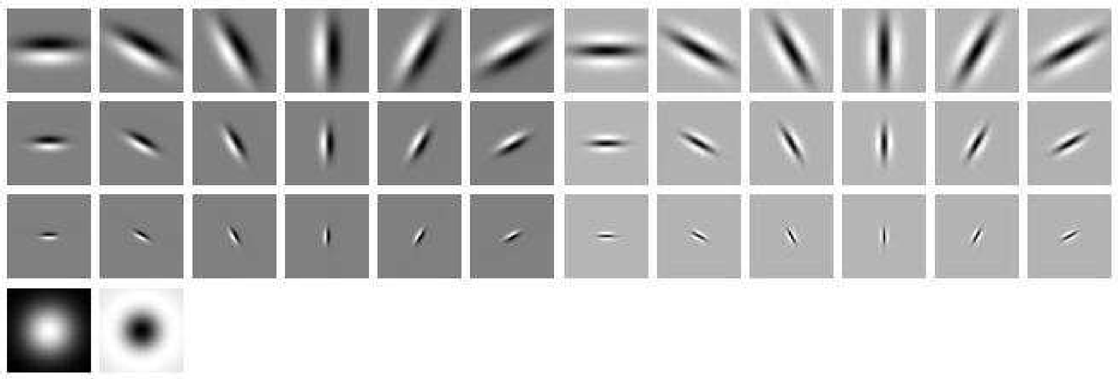


Figure 2.4.: The MR8 filter bank. The filter bank consists of 2 anisotropic filters in 3 scale and 6 orientations, a Laplacian of Gaussian and a Gaussian filter. Only 8 filters are used, by combining the 2 isotropic filters with the maximum response filters across all orientations.

2.3. Vocabulary Construction

The vocabulary construction procedure uses the filter response vectors that were obtained in an earlier stage, where filter banks were convolved with the images in a collection. Based on the assumption that numerous prototype filter responses exist, of which all others are noisy variations, we can construct a visual vocabulary by clustering all the filter responses. Eventually the visual vocabulary will consist of all the acquired cluster centers, which correspond to the prototype image features.

The vocabulary construction process contains a lot of parameters that can be applied differently, such as size of the vocabulary, clustering methods, dimensionality of descriptors, late- or early-fusion of filter responses, and descriptor-to-visual-word correspondence. In the following sections we discuss some of the choices that can be made throughout the process of vocabulary construction.

2.3.1. Fusion of Filter Responses

One of the distinctions in the vocabulary building process, is the fashion by which the filter responses are fused. One possibility would be to construct a single vocabulary using the response vectors of all images in the dataset together, without making a distinction between object classes. In this case we simply construct vocabulary $\mathcal{T}$ with size T in a straightforward manner. Clustering is performed on the response vectors of all images in the dataset. We will refer to this as *early fusion*. Early fusion is illustrated in figure 2.5.

Another possibility is to construct multiple vocabularies, one for each class, using the response vectors of each separate object class. In this case we construct multiple vocabularies $\mathcal{T}_1, \cdots, \mathcal{T}_n$ with size $\frac{T}{n}$ and conjoin them to form $\mathcal{T}$. We will refer to this as *late fusion*. Late fusion is illustrated in figure 2.6.

The difference between early fusion, illustrated in figure 2.5 and late fusion, illustrated in figure 2.6, is twofold. The first difference between early and late fusion is the position of the fusion within the vocabulary construction process. Early fusion is applied before clustering is performed, while late fusion is applied afterwards.

Aside for the difference in position, early and late fusion differ in the type of information that is fused. Early fusion concatenates low level information, that is to say the filter responses. Late fusion concatenates higher level information, namely the visual class vocabularies that are a results of clustering the response vectors of each class separately.

Needless the say, we expect the classification results for both types of fusion to be different. It is to be expected that the late fusion approach will perform the best. The reason that we expect late fusion to outperform early fusion, is that late fusion will find specific visual words, for each of the various classes. This can be compared to describing images of different object classes in a distinct jargon. Early fusion however, will just result in a visual vocabulary that aggregates across all object categories. The images can only be distinguished by the co-occurrence or frequency of specific words. On the other hand, since the late fusion approach construct severals vocabularies independently, there is a risk of finding redundant words, i.e. finding similar words in each of the class vocabularies. This would limit the expressiveness of the resulting visual vocabulary.

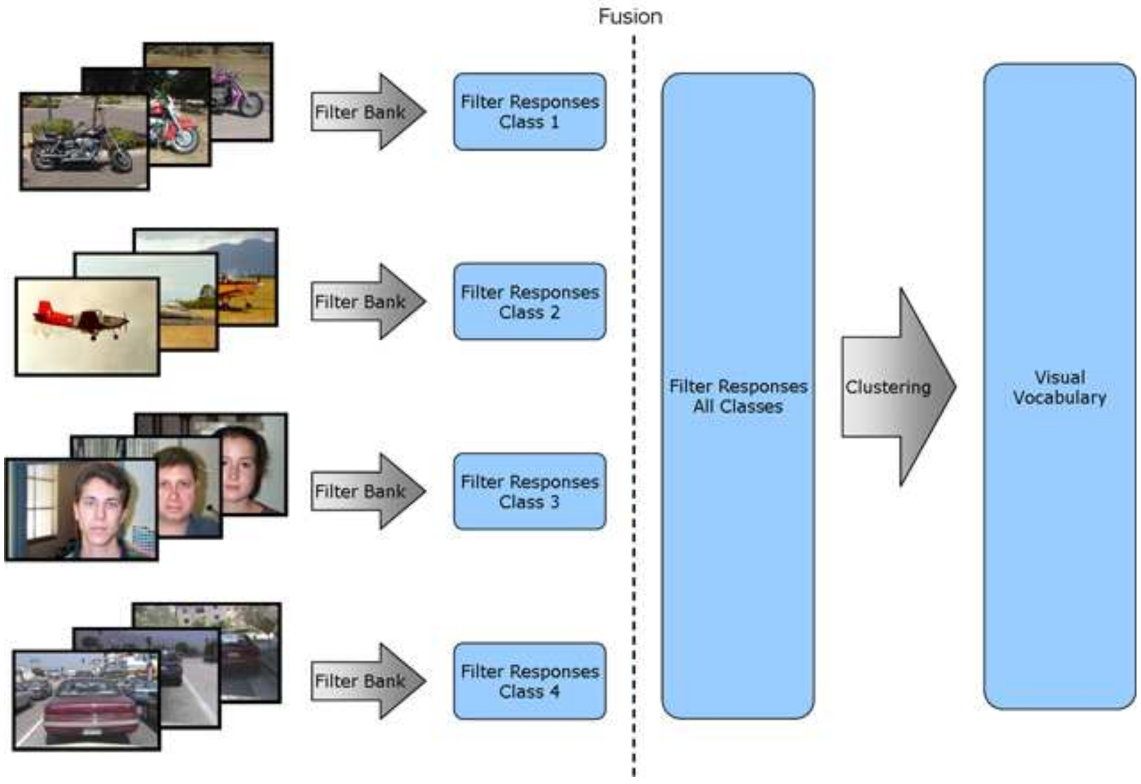


Figure 2.5.: Early fusion of filter responses. First response vectors are extracted using a filter bank. Subsequently, filer responses are concatenated before clustering is performed. The position in the process where fusion is applied, is indicated by a dotted line.

2.3.2. Clustering

The vocabulary construction process basically boils down to clustering. In our case, we will be clustering the response vectors produced by convolving a set of images with a filter bank. By clustering these response vectors, we determine the appropriate prototype response vectors that make up the visual vocabulary.

The aim of each clustering algorithm is to partition some dataset into several subsets that share a common trade. In most cases the similarity between instances of a particular dataset is measured by proximity according to a distance measure. There are many different clustering methods, which can be supervised or unsupervised, hierarchical or non hierarchical, optimal or semi-optimal. In this case we will focus on non hierarchical unsupervised clustering, since we are not interested in creating a hierarchy within visual words, nor do we posses labels that would be useful in the vocabulary building process[2]. There are several clustering algorithms that satisfy our requirements, such as EM-clustering k-means and x-means. We will adopt the use of k-means [18], which is commonly used for vocabulary building [7, 5, 8, 9, 17].

[2]The late fusion approach could be viewed as semi-supervised clustering, because we use the class labels to partition the dataset beforehand. However, the actual clustering process does not make use of labels.

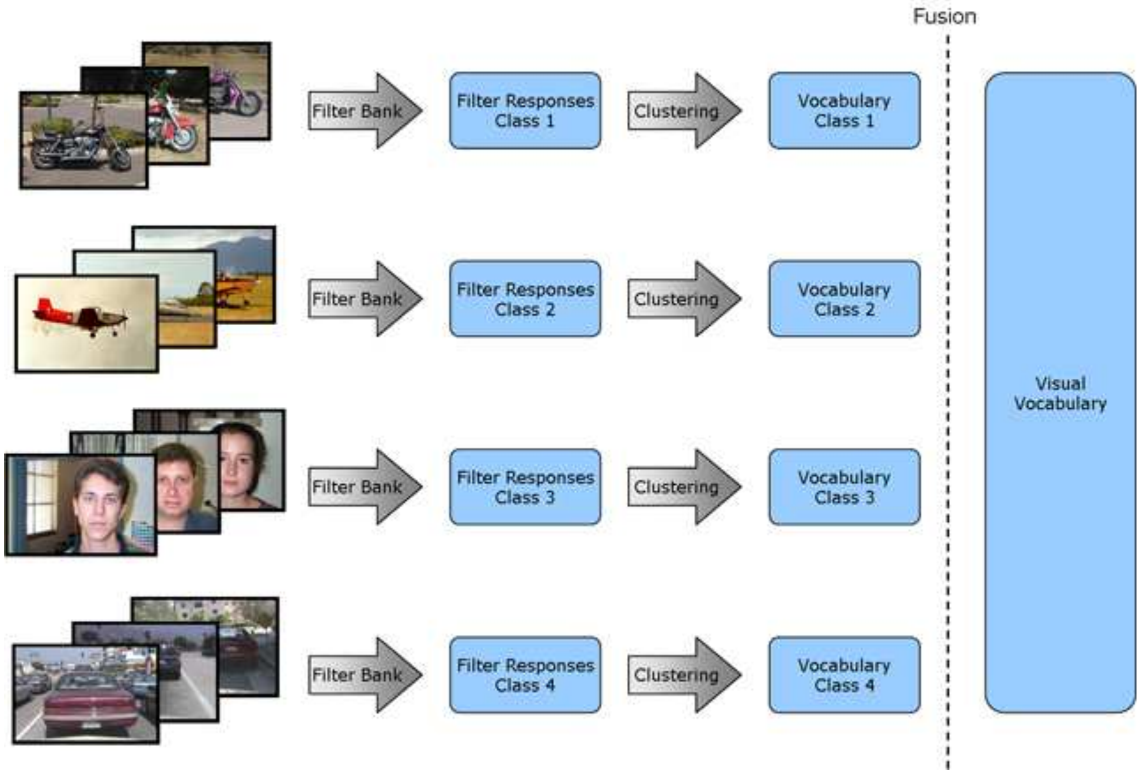


Figure 2.6.: Late fusion visual class vocabularies. First response vectors are extracted using a filter bank. Subsequently, clustering is performed on the responses vectors of each separate class, constructing a vocabulary for all the various classes. Finally, the vocabularies are concatenated, thus creating a single vocabulary. The position in the process where fusion is applied, is indicated by a dotted line.

K-Means Clustering

K-means clustering is a clustering algorithm where the aim is to partition the datapoints, in our case response vectors, into k clusters. The partitioning is based on minimizing the total within-group sum of squares distance, which is similar to minimizing equation (2.1).

$$\sum_{i=1}^{k} \sum_{x_j \in S_i} |x_j - \mu_i|^2 \qquad \text{, with } \mu_i \text{ mean of cluster } S_i \tag{2.1}$$

K-means clustering finds clusters in a greedy manner, by starting with an initial partition and iteratively updating the partition until convergence is met. This means that it is not guaranteed to return the global optimum, but will most likely result in a local maximum. The quality of the resulting centers is dependent on several things, such as the initialization and the natural clustering of the dataset. We are looking for exactly k clusters, so the resulting cluster centers may look strange if the dataset is poorly, or highly clustered.

2.4. Image Descriptors

Once a visual vocabulary is constructed, it is possible to determine the image descriptors. This is done by mapping the filter responses onto the visual dictionary. Mapping the filter responses onto the visual dictionary simply means assigning the corresponding visual word for each response vector, i.e. finding the visual word that minimizes some distant metric.

In line with the *bag of keypoints* approach, we unite all image descriptors of an image into one large group. By counting the number of times each visual word occurs within the image, we can construct a frequency vector. Usually this frequency vector is referred to as a descriptor histogram, as it can be visualized by a histogram. In figure 2.7 an example of such a histogram is shown.

For each image the descriptor histogram will be different, but the assumption is that throughout the image categories, these histograms show similar patterns for images of the same class, and dissimilar patterns for images of different classes.

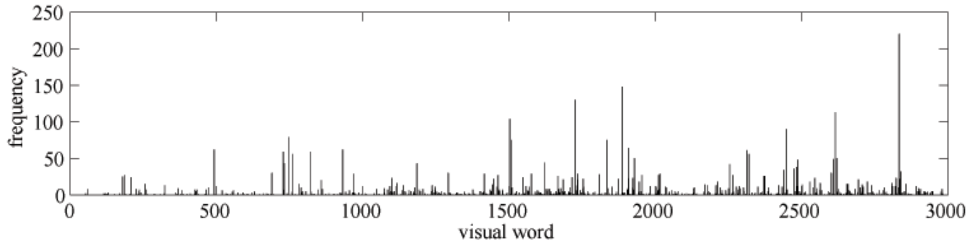


Figure 2.7.: Example of a descriptor histogram.

Due to the large difference in the number of response vectors for each image, we need to normalize the descriptor histograms. This is necessary to insure that the descriptor histograms are within the same spatial domain. Normalization is performed by dividing each of the histogram bin entries by the total amount of histogram bin entries, using equation (2.2). As a result the total sum over the histogram bins will be 1, and each histogram bin will have a value between 0 and 1.

$$H_j = \frac{H_j}{\sum_{i=1}^{T} H_i} \tag{2.2}$$

where H_i and H_j are the i^{th} and j^{th} bin of the descriptor histogram and T is the dictionary size.

Chapter 3

Object Classification

Object classification addresses the problem of discriminating between object classes. Images should be labeled according to their corresponding class. Formally, we could say that given an image I, a set of object categories Ω, the goal is to find the object category $\omega \in \Omega$ that corresponds to the object depicted in I.

However, classification, also know as statistical pattern recognition, does not limit itself to object classification alone, and is a very active area of research. Statistical pattern recognition aims to classify datapoints based on either prior knowledge of the classification problem or on statistical information extracted from the dataset. The points to be classified, are usually groups of measurements or observations, defined in some multidimensional space. In our case the datapoints are the image descriptor defined in the former chapter.

Formally, classification of data can be described as follows. Given a dataset D and a collection of classes Ω, the aim is produce a classifier $h : x \rightarrow \omega$, which maps each instance $x \in D$, to the correct class $\omega \in \Omega$.

In general, there are two main approaches to classification. We could either define a generative model, in which case we model a class by a distribution over its descriptors, or learn a discriminative model for the data, in which case we try to estimate a decision boundary. In both cases we assume that the distribution of various classes are sufficiently different, to make accurate classification possible.

Generative classification methods, rather than producing a label directly, estimate the conditional class probability $p(x|\omega)$. In combination with the prior probability of a particular class $p(\omega)$ the classifier h can be constructed. The decision boundary can be determined using Bayes rule for minimum error, defined by equation (3.1).

$$\forall j, k \; p(x|\omega_j)p(\omega_j) > p(x|\omega_k)p(\omega_k) \qquad \omega_k \in \Omega \text{ and } k \neq j \tag{3.1}$$

Several methods can be applied to produce an estimate for $p(x|\omega)$, of which some will be discussed in this chapter.

In this chapter several well known classification methods are described, as well as some

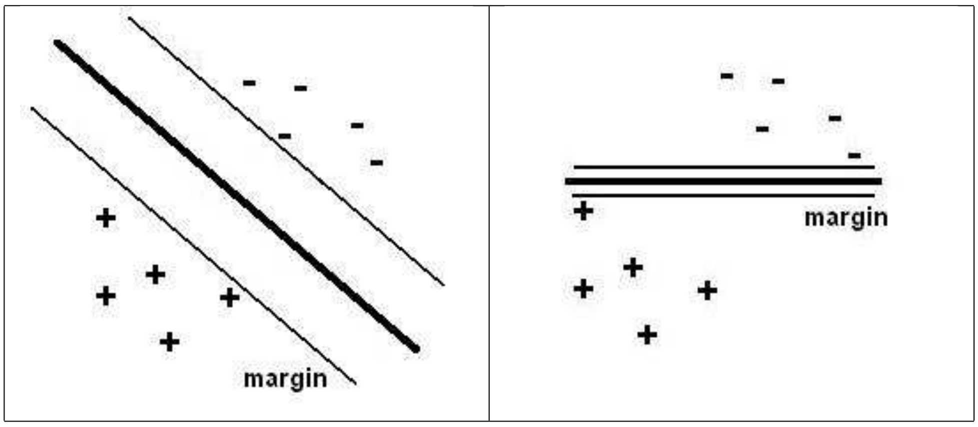


Figure 3.1.: Example of a linear separable dataset and multiple decision boundaries. The figure on the left has a superior decision boundary to the figure on the right, because the margin are greater. In fact, the figure on the left shows the optimal decision boundary, where the distances between the decision boundary and the closest points are maximized.

methods that are less conventional. The methods that will be discussed in the following subsections are support vector machines (SVMs), naive Bayes, and a simplified version of the naive Bayes model (Bhattacharyya distance).

3.1. Support Vector Machines

Support vector machines (SVMs) where first introduced by Vladimir Vapnik in 1995 [19], and it remains a widely applied classification technique today. One major advantage of SVMs is the fact that it performs very well with high dimensional data, as well as data that is sparse. The vocabulary size we will be using is in the order of thousands, and the number of training samples in the order of several hundreds, so the dataset is both sparse and highly dimensional. Thus, SVMs would be an appropriate classification method in this case

In contrast to the other methods in this section, SVMs do not estimate the class conditionals. Instead of using the Bayes rule of minimum error to determine a decision boundary, the aim of SVMs is to determine the decision boundary directly. In addition to finding a decision boundary that partitions the data correctly (there might be many solutions), we are interested to find the boundary that achieves maximum separation between data of different classes. The vectors that lie closest to the decision boundary are called support vectors.

Before considering N-dimensional hyperplanes, let's look at the simple 2-dimensional problem shown in figure 3.1. The aim is to find a linear discriminant that will partition data. In this case the data is linearly separable, meaning that all the vectors labeled with $+$ are situated at one side of the line and the vectors labeled with - on the other side. There are an infinite number of possible lines that separate the two classes, so the optimal solution will be the line that maximizes the margins between the decision boundary and the support vectors.

For data that is not linearly separable, we will make use of the 'kernel trick'. The original data is mapped to another space, the feature space F, applying a Mercer kernel operator K. Where normally the hyperplane would be described by

$$f(\mathbf{x}) = a\mathbf{x} + b \tag{3.2}$$

now we can write

$$f(\mathbf{x}) = \sum_{i=1}^{n} a_i K(x_i) + b \tag{3.3}$$

With use of the kernel we can map the data to feature space F and the support vectors now correspond to the hyperplane, that separates the data with maximal margins in this feature space. Through use of a kernel we can define more complex boundaries than ordinary linear boundaries. For data which is N-dimensional the same trick can be applied.

Any kernel can be applied, but only a few are frequently used. Among the most frequent are the linear kernel, the polynomial kernel, the sigmoid kernel and the radial basis function (RBF) kernel. Each of the kernels have their own set of parameters, and naturally these have to be optimized for maximal classification results.

A drawback of SVMs is that they are only capable of binary classification. This implies that an SVM can only discriminate between two classes. In most cases an SVM is used as a classifier for just one class, by classifying whether or not an instance is part of a particular class. To adapt SVMs to handle multiclass problems there are several options, of which I will only mention a few. The first option is multiclass ranking SVMs, in which one SVM decision function attempts to classify all classes. The aim is to find a single mathematical function that separates all classes. In most multiclass problems this approach is unreliable, since a mathematical function that separates all classes does not exist. A second option is pairwise classification, in which there is one binary SVM for each pair of classes to separate members of one class from members of the other class. It is expected that this approach will be slower than other approaches, since the number of SVMs that need to be learned will be greater. Yet another approach is one-against-all classification, in which there is one binary SVM for each class to separate members of that class from members of all other classes. This means that a single SVM is learned for each of the various classes. In most cases, as well as in our situation, the last approach is preferred.

3.1.1. RBF Kernel

One of the kernels we will apply in this project is the RBF kernel, defined by equation (3.4). It maps data to the feature space with use of multiple radial basis functions. You could say that the decision boundary is constructed by placing radial basis functions over the support vectors within the normal space.

$$K(u,v) = \exp\left(-\gamma * |u-v|^2\right) \tag{3.4}$$

where γ determines the RBF width.

3.1.2. Probabilistic Outputs

SVMs do not provide a probability output, but provide the classifier h directly. If we want to classify instances in a similar manner as all the other methods, we need to map the SVM outputs to probabilities. John Platt [20] showed that we can do just this, by training an sigmoid using the SVM outputs, which provides us with probabilities. The sigmoid function is shown in equation (3.5).

$$P(y=1|f) = \frac{1}{1+\exp\left(Af+B\right)} \tag{3.5}$$

where A and B are the sigmoid parameters that need to be fitted and f is the output of the trained SVM.

3.2. Naive Bayes

An object model that was used by Winn [7] was specifically designed for the purpose of object classification using a visual vocabulary and descriptor histograms. The set of descriptor histograms for each class are modelled through the use of a Gaussian distribution with mean $\bar{\mathbf{H}}_\omega$ and diagonal covariance whose diagonal entries form the vector β_ω. Due to the utilization of just the diagonal of the covariance matrix, all histogram bins are considered independent. This assumption is typical for naive Bayesian models.

The probabilistic relationship for a class with parameters $\theta = (\bar{\mathbf{H}}, \beta)$ is defined as follows.

$$p(\mathbf{H}|\theta) = \prod_{i=1}^{T} \mathcal{N}(H_i^{\frac{1}{2}}, \bar{H}_i^{\frac{1}{2}}, \beta_i) \tag{3.6}$$

where H_i is the i^{th} bin of the descriptor histogram and T is the dictionary size. The value of each bin is raised to the power of a halve, effectively making the variance constant, rather then linearly dependent on the mean $\bar{\mathbf{H}}$ [21].

One of the key assumptions of this method is that when an object of class ω appears in the image, the corresponding histogram $\mathbf{H}$ of the region in which the object lies, is close to the class histogram $\bar{\mathbf{H}}_\omega$ in terms of Mahalanobis distance given by β_ω.

If we take a closer look at the logarithm of $P(\mathbf{H}, \theta)$ we can see the resemblance of this approach to the Bhattacharyya distance.

$$log\left[P(\mathbf{H}|\theta)\right] = \tag{3.7}$$

$$log\left[\prod_{i=1}^{T} \mathcal{N}(H_i^{\frac{1}{2}}, \bar{H}_i^{\frac{1}{2}}, \beta_i)\right] = \tag{3.8}$$

$$\sum_{i=1}^{T} log\left[\mathcal{N}(H_i^{\frac{1}{2}}, \bar{H}_i^{\frac{1}{2}}, \beta_i)\right] = \tag{3.9}$$

$$\sum_{i=1}^{T} log\left[\frac{1}{\sqrt{\beta_i}\sqrt{2\pi}} exp^{-\frac{(H_i^{\frac{1}{2}} - \bar{H}_i^{\frac{1}{2}})^2}{2\beta_i}}\right] = \tag{3.10}$$

$$\sum_{i=1}^{T} log\left[(2\pi\beta_i)^{-\frac{1}{2}}\right] - \frac{(H_i^{\frac{1}{2}} - \bar{H}_i^{\frac{1}{2}})^2}{2\beta_i} = \tag{3.11}$$

$$\sum_{i=1}^{T} -\frac{1}{2} log\left[2\pi\beta_i\right] - \frac{(H_i^{\frac{1}{2}} - \bar{H}_i^{\frac{1}{2}})^2}{2\beta_i} = \tag{3.12}$$

$$\sum_{i=1}^{T} -\frac{1}{2} log\left[2\pi\right] - \frac{1}{2} log\left[\beta_i\right] - \frac{(H_i^{\frac{1}{2}} - \bar{H}_i^{\frac{1}{2}})^2}{2\beta_i} \tag{3.13}$$

If we remove the constants and work out further we get.

$$\sum_{i=1}^{T} -\frac{1}{2} log\left[\beta_i\right] - \frac{1}{2}\frac{\sqrt{H_i}^2}{\beta_i} - \frac{1}{2}\frac{\sqrt{\bar{H}_i}^2}{\beta_i} + \frac{\sqrt{H_i \bar{H}_i}}{\beta_i} = \tag{3.14}$$

$$\sum_{i=1}^{T} -\frac{1}{2} log\left[\beta_i\right] - \frac{1}{2}\frac{|H_i|}{\beta_i} - \frac{1}{2}\frac{|\bar{H}_i|}{\beta_i} + \frac{\sqrt{H_i \bar{H}_i}}{\beta_i} \tag{3.15}$$

When β is assumed to be constant, and the histograms are normalized, equation 3.15 reduces to the Battacharyya coefficient (equation 3.16), due to the fact that all other parts of the equation become constant. When β is not constant, this method differs from the Bhattacharyya distance, because it normalizes all vector space axes using variance.

3.3. Bhattacharyya Coefficient

The Battacharyya coefficient is not a method for classification, but rather a similarity measure, which measures the similarity of two discrete probability distributions. It is widely applied in the field of histogram tracking, where it is used to calculate the similarity between two histograms [22]. However, it can also be applied for classification, where it boils down to a simplified version of the formally introduced generative model.

In text categorization applications, a similar measure is used to classify text, the cosine similarity [23, 24]. In this case, the *bag of words* framework is applied, and documents are represented by frequency vectors, which indicate the number of times certain terms are used in a document. Given a set of documents D and their corresponding vector representations, the centroid is determined by averaging over the vectors in D. Once a centroid is determined for each category, the similarity between a new document and a category can be used to determine whether or not a document belongs to a certain category.

In previous sections we discussed that images can be represented by descriptor histograms. We can measure the similarity of two descriptor histograms using Bhattacharrya distance. Similar to text categorization, we can try to classify images using the centroid, or mean, to classify new instances. Given the mean histogram of a class $\bar{\mathbf{H}}_\omega$, and a particular instance H, the Bhattachayya distance can be determined as follows.

$$B(H, \bar{H}_\omega) = \sum_{i=1}^{T} \sqrt{H_i \bar{H}_{\omega,i}} \tag{3.16}$$

where H_i is the i^{th} bin of the descriptor histogram.

When the histograms are normalized, the values for the Bhattacharyya distance will always range between 0 and 1. For this reason we can apply the Bayes rule for minimum error for classification, defined by equation (3.1) in the beginning of this chapter.

Chapter 4

Object Localization

The last chapter discussed only one part of this project, object classification. This chapter will focus on the second part of the project, object localization. Currently a lot of researchers are interested in this problem [15, 10, 4, 13, 14, 11, 25]. In this chapter two approaches to object localization will be discussed. First we discuss the most commonly used approach, the tile-based approach. Furthermore a second, more sophisticated method, will be discussed, that will be compared to the tile-based approach.

Object localization can be formally described as follows. Given an image I, and a set of object classes Ω, the goal is to localize all objects depicted in I that belong to either of the object classes $\omega \in \Omega$. A localization algorithm should give a set of labeled regions, given an input image. If none of the object classes are present in the image, the algorithm should return an empty set.

4.1. Tile-Based Approach

The tile-based approach is one of the most common approaches to object localization. Basically, the procedure consists of three steps. First, we select a collection of local image patches, by sliding a prototype image patch (tile) across the image. Subsequently, every image patch is evaluated using the same classifiers we learned in the classification procedure. The classifiers provide us with the class conditional probabilities of every image patch, which in turn are used to select the final detections. Figure 4.1 shows an illustration of the tile-based approach.

The selection process has many variants. We could use a single tile, multiple tiles at different scales, or use random tiles. Furthermore, we can evaluate at each location, evaluate at random positions, or evaluate on a predefined grid. Some variants use contextual information to select certain image patches [10]. Another option would be to evaluate every possible local image patch, however this is very impractical, since the increase of patches will lead to an increase of computational complexity.

One of the disadvantages of the tile-based approach is that the image patches are selected arbitrarily, and the final detection is either one of these arbitrary image patches, or a

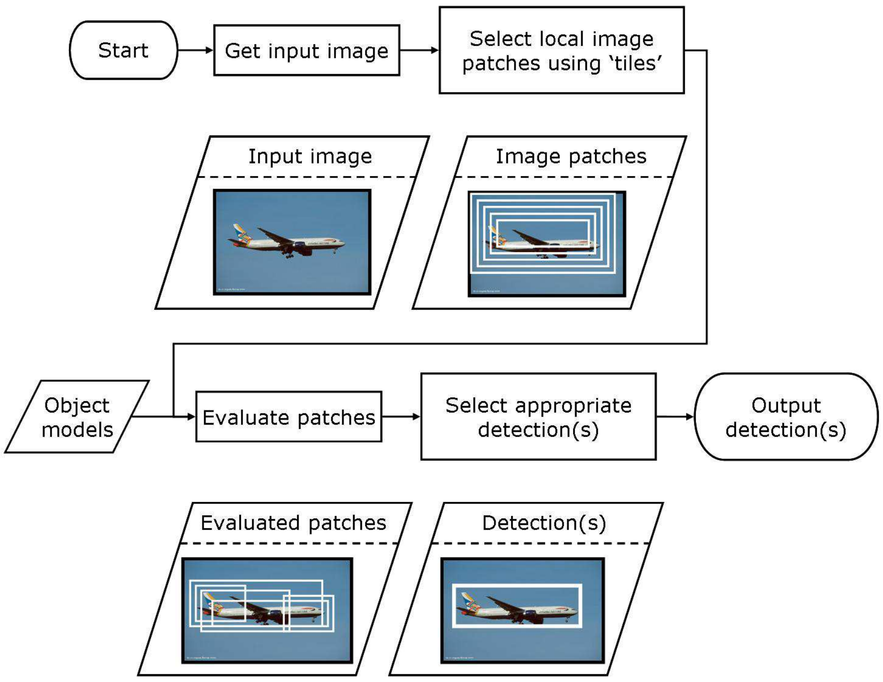
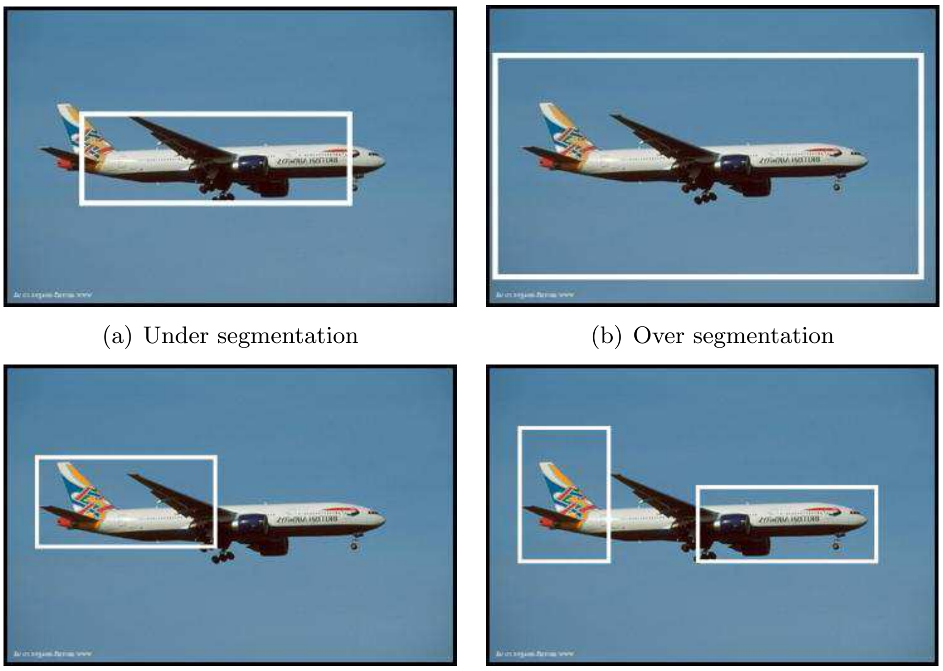


Figure 4.1.: Overview of the baseline method for object localization.

combination of image patches. As a result, the probability that the final detection will be a perfect boundingbox around an object is small, and we are likely to end up with detections that contain only a partial object, as in figure 4.2(a). On the other hand, the opposite could also happen. We could also end up with an object region with the whole object including a lot of background pixels, shown in figure 4.2(b). Other problems will arise when evaluation regions are labeled incorrectly. When parts of some object classes resemble parts of other classes, misclassification could be a possibility. Examples are shown in figure 4.2(c) and 4.2(d). The figures show what can happen when some of the evaluation regions are misclassified. One of the possible outcomes is depicted in figure 4.2(c), where only the rear of the airplane is classified correctly, resulting in a poor segmentation. Another outcome is shown in figure 4.2(d), where multiple object regions are detected.

4.2. Localization by Region Optimization

The procedure for object localization by region optimization is as follows. We start by segmenting the target image into several equally sized regions. However, in contrast to the former approach, these regions do not act as building blocks for the final detections. In stead, we use these regions as starting points for further investigation. Each of the regions acts as a seed for the optimization algorithm and we will try to optimize the similarity of a region with respect to the learned object class models. The aim of the optimization process is to maximize the similarity of a region given an object class model, with respect to the region size, shape, and location.



(a) Under segmentation

(b) Over segmentation

(c) Partial detection

(d) Detection of multiple instances

Figure 4.2.: Examples of bad segmentation. In figure (a) an example of under segmentation is shown. The object region contains only part of the total object. Figure (b) shows an example of over segmentation. Here, the object region contains the total object. However, a lot of background pixels are also present. Figure (c) shows an example where only halve of the object is detected. In figure (d) multiple instances are detected incorrectly.

As was described in chapter 2, we have adopted a visual vocabulary framework. Within this framework, image regions are defined by words from a visual vocabulary. Each pixel within an image can be mapped to exactly one visual word. This means that each region will have its own set of visual words and will, to some degree, differ from other regions. An image patch is described by a descriptor histogram, which is determined from its set of visual words. In a descriptor histogram each histogram bin corresponds to a specific word from the visual vocabulary. For an image region, the descriptor histogram is a distribution of visual words present in the region. Given the fact that each pixel of an image corresponds to exactly one visual word, we are able to apply histogram tracking methods to update certain starting regions in order to find regions with maximum potential.

Histogram tracking is normally applied for tracking of objects, nonetheless we will use histogram tracking for object localization. Object localization is not that different from object tracking, since in both cases we are trying to localize certain object within an image, given a target description. In histogram tracking applications, regions are updated by maximizing a particular similarity measure. The similarity measure determines the similarity between target model and a candidate model. In this case the target model will consist of a learned object class model and the candidate model will consist of a descriptor histogram extracted from an image patch. Not all object class models can be applied so we will only use the simplified naive Bayes model discussed in section 3.3.

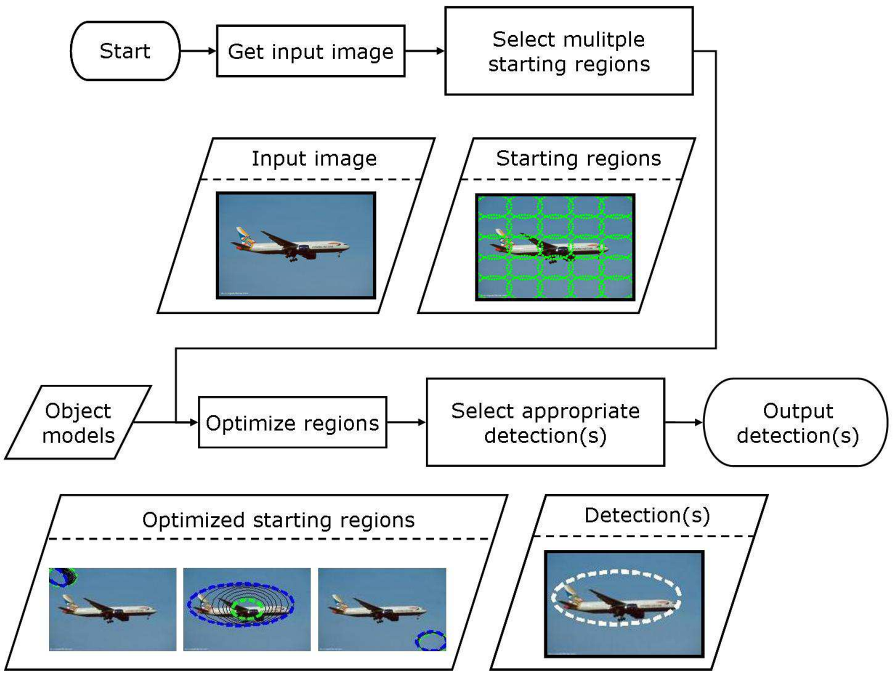


Figure 4.3.: Overview of the localization procedure using histogram tracking.

Localization of objects using histogram tracking will consist of iteratively updating particular regions by increasing the similarity measure with respect to their size, shape and location. Figure 4.3 illustrates the localization procedure using histogram tracking. In the next sections we will explain histogram tracking in detail.

The histogram tracking approach to object localization tries to deal with the problems of the tile-based approach. Where the problem of an imperfect fit is inherent to the tile-based approach, it is not inherent to the following approach. The reason being that we do not use arbitrary regions as building blocks for object regions, but rather try to optimize certain regions by changing their position shape and scale. Thus the regions in the tile-based approach are static, where the regions in this approach are dynamic. Object regions are determined by optimizing certain starting regions. The starting regions are optimized with respect to shape, size and location using histogram tracking.

4.3. Histogram Tracking

Histogram tracking is a widely applied method for object tracking. It was first introduced in [22] and remains the preferred method for tracking objects today due to the fact that it has low computational costs, so real-time applications are possible.

The first step needed for histogram tracking is to describe our target model. This model defines the characteristics of the object that we are looking for in the images and is defined by a probability density function (pdf) in the feature space. In previous sections we already discussed that images are characterized by their visual words, which can be represented by a descriptor histogram. For simplicity we will go into object class characterization later.

For now we will focus on histogram tracking of specific objects through successive frames of video.

We can characterize the target model $\mathbf{q}$ in the first frame by its pdf in feature space. In our case, the feature space will be the filter response space, and the pdf will be represented by a descriptor histogram, which in turn is a mapping from the feature space to a discrete distribution within the feature space. We also define the location of the target model, which is set to 0. In the next frame a target candidate is defined at location $\mathbf{y}$, and characterized by the pdf $\mathbf{p}(\mathbf{y})$. In our case this pdf will be the descriptor histogram of the region defined by the location $\mathbf{y}$ and its surrounding pixels.

To compare the target model with the target candidate, we need a similarity function, which is defined by

$$\rho(\mathbf{y}) \equiv \rho[\mathbf{p}(\mathbf{y}), \mathbf{q}] \tag{4.1}$$

The similarity function $\rho(\mathbf{y})$ is the function that needs to be maximized, and its maximum indicates objects in the second frame that have similar representation to $\mathbf{q}$ which was defined in the first frame. Due to the fact that we are using histograms as a representational model, all spatial information is lost. Spatial information is crucial if we want to determine in what direction we should focus our search. To regain some spatial information, an isotropic kernel in the spatial domain is used as a mask over the target object. By using a kernel, the function $\rho(\mathbf{y})$ becomes a smooth function in $\mathbf{y}$, which enables us to perform gradient based optimization methods.

Let us redefine a target model, where the kernel is taken into account. All targets are normalized to a unit circle, to eliminate target dimensions, by rescaling the row and column dimensions.

The normalized pixel locations in the region of the target model are described by $\{\mathbf{x}_i^*\}_{i=1\cdots n}$. The region is centered at 0 in the first frame.

The function $b : R^2 \to \{1 \cdots T\}$ is a mapping from the pixel on location $\mathbf{x}_i^*$ to the index $b(\mathbf{x}_i^*)$ of its histogram bin, i.e. the corresponding visual word. The probability that a visual word $u = 1 \cdots T$ will be present in the target model can now be written as

$$q_u = C \sum_{i=1}^{n} k\left(||\mathbf{x}_i^*||^2\right) \delta\left[b(\mathbf{x}_i^*) - u\right] \tag{4.2}$$

where δ is the Kronecker delta function. C is a normalization constant, which is define by

$$C = \frac{1}{\sum_{i=1}^{n} k\left(||\mathbf{x}_i^*||^2\right)} \tag{4.3}$$

The kernel can be incorporated into the target candidate as follows. Let $\{\mathbf{x}_i\}_{i=1\cdots n_h}$ be the normalized pixel locations of the target candidate. The region is centered at y in the second frame. A similar kernel is used in the second frame, only a bandwidth parameter h is introduced. The probability that a visual word $u = 1 \cdots T$ will be present in the target candidate is given by

$$p_u(\mathbf{y}) = C_h \sum_{i=1}^{n_h} k\left(\left\|\frac{\mathbf{y}-\mathbf{x}_i}{h}\right\|^2\right)\delta\left[b(\mathbf{x}_i)-u\right] \tag{4.4}$$

where the constant C_h in defined by

$$C_h = \frac{1}{\sum_{i=1}^{n_h} k\left(\left\|\frac{\mathbf{y}-\mathbf{x}_i}{h}\right\|^2\right)} \tag{4.5}$$

4.3.1. Target Candidate Localization

Locating the target object in the second frame is equal to maximizing the similarity function $\rho\left[\mathbf{p}(\mathbf{y}),\mathbf{q}\right]$ with respect to $\mathbf{y}$. To locate the target candidate, we start from the position of the target model in the previous frame $\mathbf{y}_0$ and search in its vicinity. As was mentioned earlier, the use of a kernel smoothens the pdf $\mathbf{p}(\mathbf{y})$ in $\mathbf{y}$, which enables us to perform gradient based optimization methods. We will apply the mean shift procedure [26] for this purpose.

In section 3.3 we used the Bhattacharyya coefficient to classify object. Here we will use the Bhattacharyya coefficient for the similarity function $\rho\left[\mathbf{p}(\mathbf{y}),\mathbf{q}\right]$. The Bhattacharyya coefficient is given by

$$\rho\left[\mathbf{p}(\mathbf{y}),\mathbf{q}\right] = \sum_{i=1}^{T}\sqrt{p_u(\mathbf{y})q_u} \tag{4.6}$$

We can approximate the Bhattacharyya coefficient by using Taylor expansion around $\mathbf{y}_0$. The linear approximation of the Bhattacharyya coefficient is

$$\rho\left[\mathbf{p}(\mathbf{y}),\mathbf{q}\right] \approx \frac{1}{2}\sum_{u=1}^{T}\sqrt{p_u(\mathbf{y}_0)q_u} + \frac{1}{2}\sum_{u=1}^{T} p_u(\mathbf{y})\sqrt{\frac{q_u}{p_u(\mathbf{y}_0)}} \tag{4.7}$$

An approximation will suffice when the target candidate $p_u(\mathbf{y})$ is not rigorously different from $p_u(y_0)$. For successive frames in a video stream this is not an unrealistic assumption. If we look at equation 4.4, the equation above can be reduced to

$$\rho\left[\mathbf{p}(\mathbf{y}),\mathbf{q}\right] \approx \frac{1}{2}\sum_{u=1}^{T}\sqrt{p_u(\mathbf{y}_0)q_u} + \frac{C_h}{2}\sum_{i=1}^{n_h} w_i k\left(\left\|\frac{\mathbf{y}-\mathbf{x}_i}{h}\right\|^2\right) \tag{4.8}$$

with

$$w_i = \sum_{u=1}^{m}\sqrt{\frac{q_u}{p_u(\mathbf{y}_0)}}\delta\left[b(\mathbf{x}_i)-u\right] \tag{4.9}$$

Finally, to maximize the similarity $\rho\left[\mathbf{p}(\mathbf{y}),\mathbf{q}\right]$ we should maximize equation 4.8 with respect to $\mathbf{y}$. In order to do so, we only need to maximize the second part of the equation, because the first part is independent of $\mathbf{y}$.

We can employ the mean shift procedure to produce the sought location [26]. It finds the new location by moving along the gradient of the density estimate. It recursively updates the current location $\mathbf{y}_0$ to a new location $\mathbf{y}_1$ through use of

$$y_1 = \frac{\sum_{i=1}^{n_h} \mathbf{x}_i w_i g\left(\left\|\frac{\mathbf{y}_0 - \mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n_h} w_i g\left(\left\|\frac{\mathbf{y}_0 - \mathbf{x}_i}{h}\right\|^2\right)} \tag{4.10}$$

where $g(\mathbf{x}) = -k(\mathbf{x})$.

4.3.2. Practical Algorithm for Localization

In contrast to object tracking, we are not dealing with images that change through time and are not interested tracking object within a changing environment. Instead we are interested in locating objects within static images. However, histogram tracking can be applied, since it is nothing more than maximizing some pdf with respect to location, which is exactly what we are trying to do.

The same way we used the Bhattacharyya coefficient for classification purposes in section 3.3, we can use it to localize objects. The target model will be the centroid or mean of an object category. The target candidate will be the descriptor histogram of an image region. The starting image region will be selected through use of the procedure in section 4.2. The pdf will be maximized through multiple position updates, until convergence is met. Pseudo code for the procedure is given in algorithm (1).

Algorithm 1: Mean shift procedure for object tracking. Steps four through six are to compensate for the linear approximation of the Battacharyya coefficient, using Taylor expansion.

```
   Input: Target model $\mathbf{q} = \{q_u\}_{u=1\cdots T}$, and its location $\mathbf{y}_0$
   Output: New location $\mathbf{y}_1$
1  initialize the location of the target in the current frame with $\mathbf{y}_0$, compute
   $\{p_u(\mathbf{y}_0)\}_{u=1\cdots T}$ and evaluate $\rho[\mathbf{p}(\mathbf{y}_0), \mathbf{q}]$
2  determine $\{w_i\}_{i=1\cdots n_h}$ according to (4.9)
3  determine location $\mathbf{y}_1$ according to (4.10)
4  while $\rho[\mathbf{p}(\mathbf{y}_1), \mathbf{q}] < \rho[\mathbf{p}(\mathbf{y}_0), \mathbf{q}]$ do        /* compensation */
5      $\mathbf{y}_1 \leftarrow \frac{1}{2}(\mathbf{y}_0 + \mathbf{y}1)$
6  end
7  if $\|\mathbf{y}_1 - \mathbf{y}_0\| < \epsilon$ then
8      return $\mathbf{y}_1$
9  else
10     $\mathbf{y}_0 \leftarrow \mathbf{y}_1$
11     go to line 2
12 end
```

4.4. EM-Shift

The mean shift procedure that was discussed in the previous section, is a procedure which updates the location of a target region. However, we are not only interested in the location of an object, but also in the size and shape of the target region. The mean shift procedure is not designed to include these parameters. A simple solution would be to adapt the target region with multiple scales. Subsequently the scale that resulted in the highest similarity score would be selected. However, Kröse and Zivkovic introduced a new form of tracking, EM shift [27], which estimates the size and shape of the ellipse directly. In this work not only the position of the candidate region is approximated, but the covariance matrix of the candidate region is approximated as well.

The approach to EM shift is based on an extreme outlier model [28]. Here we define a set of N independent data samples by $X = \{\vec{x}_i, \cdots \vec{x}_N\}$. We assume that the pdf $p(\vec{x})$ can be modelled by a normal distribution, so

$$p(\vec{x}|\theta) = \mathcal{N}(\vec{x}|\theta) \tag{4.11}$$

where $\theta = \{\vec{\mu}, \Sigma\}$. The Likelihood function, which can be used to estimate the mean vector $\vec{\mu}$ and covariance matrix Σ, is denoted by

$$p(X|\theta) = \prod_{i=1}^{N} p(\vec{x}_i|\theta) \tag{4.12}$$

Some prefer to use

$$p(X|\theta) \approx \sum_{i=1}^{N} p(\vec{x}_i|\theta) \tag{4.13}$$

which can only be used in a very extreme case of outliers [28].

Outliers can be modelled by a uniform distribution $1/A$, where A is the area of the domain of $\vec{x}$. If e represents the probability that a data sample is an outlier, a generative model can be written as follows

$$p'(\vec{x}_i|\theta) = e/A + (1-e)p(\vec{x}_i|\theta) \tag{4.14}$$

The Likelihood function becomes $\prod_{i=1}^{N} p'(\vec{x}_i|\theta)$. In this case the approximation of the Likelihood function using Taylor expansion in $(1-e)$ is given by

$$(e/A)^N + (e/A)^{N-1}(1-e)\sum_{i=1}^{N} p(\vec{x}_i|\theta) + O((1-e)^2) \tag{4.15}$$

If we consider the extreme outlier model, where lots of outliers are present (e is very close to 1), only the first two terms matter, and maximizing (4.15) over θ is equivalent to maximizing (4.13). Mean shift can be used to accurately estimate $\vec{\mu}$, which we will discuss in section 4.4.1. The extreme outlier model can also be used to estimate Σ, which we will describe in section 4.4.2.

4.4.1. Estimating $\vec{\mu}$ in an EM-like Manner

A more general version of (4.13), where each data sample is weighted by weight w_i is given by

$$f(\vec{\mu},\Sigma)=\sum_{i=1}^{N} w_i\mathcal{N}(\vec{x}_i|\vec{\mu},\Sigma) \tag{4.16}$$

The aim is to find the optimal parameters for $f(\vec{\mu},\Sigma)$, thus maximizing the value of function (4.16). This is done by EM-like iterations, where each iteration the value of (4.16) is increased. The E and M steps can be found using Jensen's inequality [27]

$$\log f(\vec{\mu},\Sigma)\geq G(\vec{\mu},\Sigma,q_1,\cdots,q_N)=\sum_{i=1}^{N}\log\left[\frac{w_i\mathcal{N}(\vec{x}_i|\vec{\mu},\Sigma)}{q_i}\right]^{q_i} \tag{4.17}$$

where q_i-s are constants with the following properties

$$\sum_{i=1}^{N} q_i=1 \text{ and } q_i\geq 0 \tag{4.18}$$

The E step can be found by maximizing G with respect to $\{q_1,\cdots q_N\}$, while keeping $\vec{\mu}$ and Σ fixed. The maximum is achieved by the following equation for q_i

$$q_i=\frac{w_i\mathcal{N}(\vec{x}_i|\vec{\mu},\Sigma)}{\sum_{i=1}^{N} w_i\mathcal{N}(\vec{x}_i|\vec{\mu},\Sigma)} \tag{4.19}$$

The M step is found by maximizing G with respect to $\vec{\mu}$ and Σ, while keeping $\{q_1,\cdots q_N\}$ constant. This means we need to optimize over the part of the function G that is dependent on the parameters

$$g(\vec{\mu},\Sigma)=\sum_{i=1}^{N} q_i\mathcal{N}(\vec{x}_i|\vec{\mu},\Sigma) \tag{4.20}$$

Maximizing g with respect to $\vec{\mu}$ gives

$$\vec{\mu}=\sum_{i=1}^{N} q_i\vec{x}_i=\frac{\sum_{i=1}^{N}\vec{x}_i w_i\mathcal{N}(\vec{x}_i|\vec{\mu},\Sigma)}{\sum_{i=1}^{N} w_i\mathcal{N}(\vec{x}_i|\vec{\mu},\Sigma)} \tag{4.21}$$

4.4.2. Estimating Σ

If we would like to know what the expected value for (4.13) is, we need to define the true distribution $p^*(\vec{x})$. The expectation for (4.13) becomes

$$E[f(\vec{\mu},\Sigma)] = \int_{\vec{x}} p^*(\vec{x})\mathcal{N}(\vec{x}_i|\vec{\mu},\Sigma) \tag{4.22}$$

Due to the approximated Gaussian distribution, equation (4.22) can be seen as a smoothed version of the true distribution $p^*(\vec{x})$. If we assume that $p^*(\vec{x})$ locally can be described by a Gaussian $\mathcal{N}(\vec{x}_i|\vec{\mu}^*,\Sigma^*+\Sigma)$, the expected maximum can be found for $\vec{\mu}=\vec{\mu}^*$ and unfortunately $\Sigma = 0$. For this reason the density estimate (4.16) is multiplied by $|\Sigma|^{\gamma/2}$, thus creating a 'γ-normalized' function

$$f_\gamma(\vec{\mu},\Sigma) = |\Sigma|^{\gamma/2} f(\vec{\mu},\Sigma) \tag{4.23}$$

Under the assumption that locally the real distribution $p^*(\vec{x})$ can be described by a Gaussian, the maximum with respect to Σ is at

$$\frac{\partial}{\partial\Sigma}\frac{|\Sigma|^{\gamma/2}}{|\Sigma^*+\Sigma|^{1/2}} = 0 \tag{4.24}$$

And because $\frac{\partial}{\partial\Sigma} = |\Sigma|\left[2\Sigma^{-1} - \mathrm{diag}(\Sigma^{-1})\right]$ we get

$$\begin{aligned}\gamma|\Sigma|^{\gamma}\left[2\Sigma^{-1}-\mathrm{diag}(\Sigma^{-1})\right]|\Sigma^*+\Sigma| \\ -|\Sigma|^{\gamma}|\Sigma^*+\Sigma|\left[2(\Sigma^*+\Sigma)^{-1}-\mathrm{diag}((\Sigma^*+\Sigma)^{-1})\right] = 0\end{aligned} \tag{4.25}$$

Which means that $\gamma|\Sigma|^{-1} = (\Sigma^*+\Sigma)^{-1}$ so $\Sigma = \frac{\gamma}{1-\gamma}\Sigma^*$. A desirable property of the γ-normalised solution is that the solution is not biased.

The EM-like update steps for the γ-normalized function are similar to the update steps given in the previous section, with one exception. Function (4.20) becomes

$$g(\vec{\mu},\Sigma) = \sum_{i=1}^{N} q_i \log\left[|\Sigma|^{\gamma/2}\right]\mathcal{N}(\vec{x}_i|\vec{\mu},\Sigma) \tag{4.26}$$

When we look at $\frac{\partial}{\partial\Sigma}g(\vec{\mu},\Sigma)=0$ the update step for Σ becomes

$$\Sigma^{k+1} = \beta\sum_{i=1}^{N} q_i(\vec{x}_i-\vec{\mu}^k)(\vec{x}_i-\vec{\mu}^k)^T \tag{4.27}$$

where $\beta = 1/(1-\gamma)$.

Calculating the weights

As with the approach to histogram tracking described in section 4.3, we are interested in finding the new region in which the object lies that we are trying to track. One of the differences in the former approach and this approach is in the description of the candidate region. The former approach assumed a a circular region, described by a normal distribution with diagonal covariance matrix. In this approach the target and candidate regions are defined by an ellipsoid, where the parameter $\vec{\mu}$ denotes the location of the ellipsoid and the parameter Σ its shape.

As with the former approach, the function $b : R^2 \to \{1 \cdots T\}$ is a mapping from the pixel on location $\vec{x}_i$ to the index $b(\vec{x}_i)$ of its histogram bin. However, we do not normalize the pixel locations by rescaling the row and column dimensions. Again, the histogram are made up of T bins, one bin for each word in our visual vocabulary. The probability of visual word $u = 1 \cdots T$ in the target model is the same as in (4.2), when using a Gaussian kernel

$$o_u = \sum_{i=1}^{N} \mathcal{N}(\vec{x}_i|\vec{\mu}_0, \Sigma_0)\delta\left[b(\vec{x}_i) - u\right] \tag{4.28}$$

The candidate model is described in a similar manner

$$r_u(\vec{\mu}, \Sigma) = \sum_{i=1}^{N} \mathcal{N}(\vec{x}_i|\vec{\mu}, \Sigma)\delta\left[b(\vec{x}_i) - u\right] \tag{4.29}$$

Once more the Battacharyya coefficient is used to measure the similarity between the two models and again we will use the first order Taylor expansion around the current estimate to approximate the Battacharyya coefficient

$$\rho\left[r_u(\vec{\mu}, \Sigma), o_u\right] \approx c_1 + c_2 \sum_{i=1}^{N} w_i \mathcal{N}(\vec{x}_i|\vec{\mu}, \Sigma) \tag{4.30}$$

where c_1 and c_2 are some constants and w_i is defined by

$$w_i = \sum_{i=1}^{T} \sqrt{\frac{o_u}{r_u(\vec{\mu}, \Sigma)}}\delta\left[b(\vec{x}_i) - u\right] \tag{4.31}$$

Notice the last term of (4.30), which is in the same form as (4.16). This implies that we can use the EM-like algorithm discussed in this section to search for the local maximum of (4.30).

4.4.3. Practical Algorithm

A practical algorithm for applying the EM-like algorithm for histogram tracking is presented in algorithm 2. For each frame the procedure is repeated, and each time the updated location and shape of the ellipsoid of the current frame is used as starting parameters for the next frame.

The same algorithm can be used for localization in a similar manner. Only this time the input will be a ‘seed’ region and an object class model (see section 4.2) and the output the optimized region. In all likelihood, some of the seed regions will end up converging to image regions in which the sought for objects are located.

Algorithm 2: EM shift procedure for object tracking. Optimization of a region is performed by following the EM shift procedure. The procedure is applied until the similarity score stops increasing, at which point the new region is returned. Line six indicates the cutoff criteria, where ϵ represents a small positive number.

```
   Input: Target model $o_u$, its location $\vec{\mu}_0$ and shape $\Sigma_0$
   Output: New location $\vec{\mu}_1$ and shape $\Sigma_1$
 1 initialize the location and shape of the target in the current frame with $\vec{\mu}_0$ and $\Sigma_0$,
   compute $r_u$ using (4.29) and evaluate $\rho\,[r_u(\vec{\mu}, \Sigma), o_u]$
 2 determine $\{w_i\}_{i=1\cdots n_h}$ according to (4.31)
 3 calculate $q_i$-s using (4.19)
 4 determine location $\vec{\mu}_1$ according to (4.21)
 5 determine shape $\Sigma_1$ according to (4.27)
 6 if $\rho\,[r_u(\vec{\mu}_1, \Sigma_1), o_u] - \rho\,[r_u(\vec{\mu}_0, \Sigma_0), o_u] < \epsilon$ then
 7     return $\vec{\mu}_1$ and $\Sigma_1$
 8 else
 9     $\vec{\mu}_0 \leftarrow \vec{\mu}_1$
10     $\Sigma_0 \leftarrow \Sigma_1$
11     go to line 2
12 end
```

Chapter 5

Results

The experiments that were conducted, which evaluate the implementation of the methods discussed in chapters 3 and 4, are discussed in this chapter. First we discuss the dataset that are used throughout the experiments in section 5.1. Section 5.2 goes into some data preprocessing steps that were used. Section 5.3 discusses the vocabulary construction process. In section 5.4 we discuss the classification results and section 5.5 shows the results for the localization experiments.

5.1. Caltech4 Dataset

Throughout this project we conducted experiments using a single dataset, the Caltech4 dataset. The Caltech4 dataset was first introduced in [4]. The dataset consists of 2876 images, that are divided into 4 different object classes. The classes are airplanes, cars, faces and motorbikes. The images are of different quality, which is due to differences in size. No standardized train- and test- sets are provided. Figure 5.1 shows some typical image examples. Among the datasets that are available for computer vision and image processing research, the Caltech4 dataset is not considered to be the most challenging. Nonetheless, the Caltech4 dataset is frequently used by researchers and plays a key role in this area of research.

The reason that this dataset is considered to be less challenging than some of the others, is due to the range of image variability, which is somewhat limited. Even though the appearance of instances within each class does vary, specific image conditions tend to be fairly similar. For instance, the viewpoint and orientation of instances are similar throughout the whole dataset. If we look at figure 5.1, we can see that the images within each class have similar orientation and viewpoint. All cars are photographed from the rear, the airplanes and motorbikes from the side, and the image of faces are all frontal headshots. Another similarity between all the images of the dataset is the size of the objects. In most images the objects take up most of the image and are positioned in the center of the image. Other image conditions, such as occlusion, or truncation of objects, hardly occur. Furthermore, only one object is present in most of the pictures, with exception of the cars class. To illustrate the limited image variability in the Caltech4

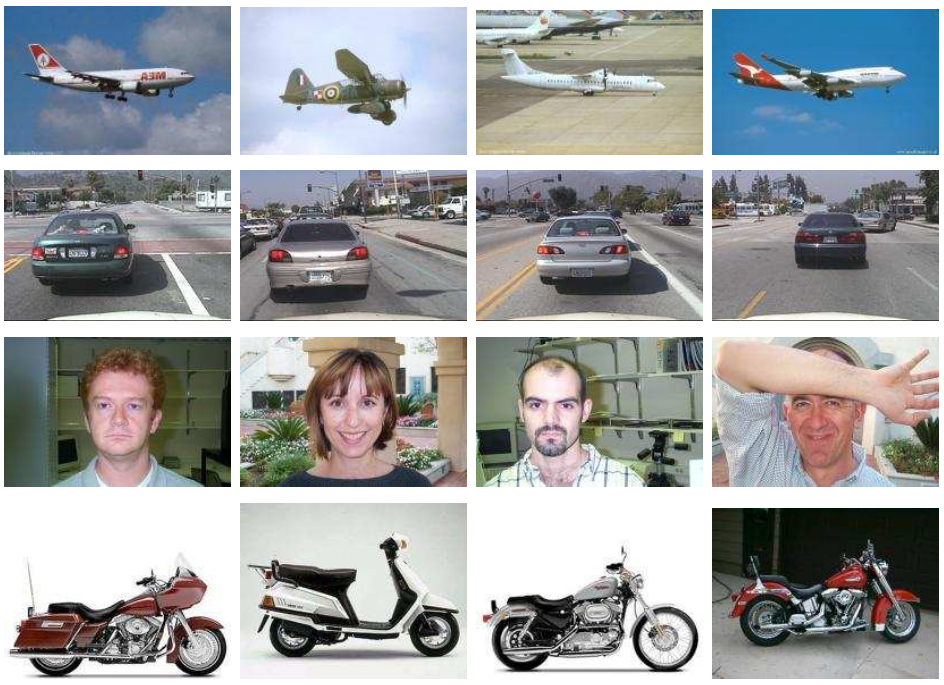


Figure 5.1.: Example images of the Caltech4 dataset.

dataset, we have constructed an average image for each of the object classes. Figure 5.2 shows images of all the classes of the Caltech4 dataset, when we average the pixel values over all the images of each separate class. We can still clearly recognize most of the four object classes. Especially the faces, motorbikes and cars classes are easily recognizable. The airplanes class is the vaguest of the four and could be mistaken for another object class, such as a boat.

The dataset is labeled using bounding boxes. The bounding boxes separate the pixels belonging to the objects from the pixels that belong to the background. Each of the bounding boxes are decoded by the coordinates of the corners of the box. Figure 5.3 shows some example images from the Caltech4 dataset, as well as examples of bounding boxes surrounding objects.

5.2. Preprocessing

In chapter 2 we introduced the visual vocabulary framework. The framework has several issues that need to be considered. In this section we will present these issues and we will discuss some of the possible solutions.

One of the disadvantages concerning the chosen approach is computational costs. During the descriptor histogram extraction procedure, several computational expensive processes are executed. First, we need to convolve a filter bank with each of the images. This means, that at each pixel location, several image filters need to be applied. Subsequently, the response vectors extracted by the convolution process, are compared to a collection of other vectors (the visual vocabulary), in order to assign each response vector to a visual

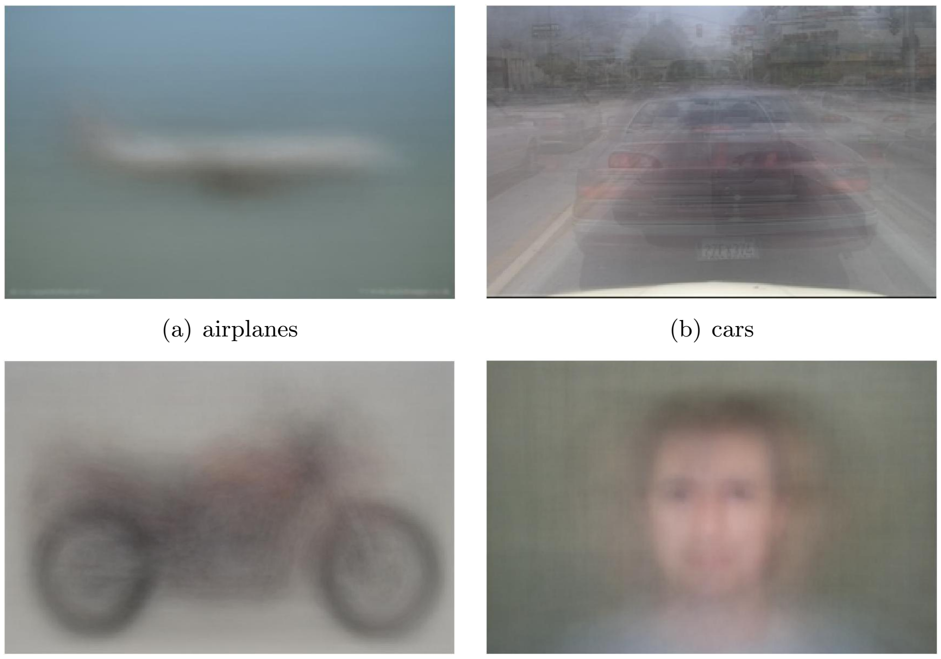


Figure 5.2.: Examples of the average image for each of the classes of the Caltech4 dataset.

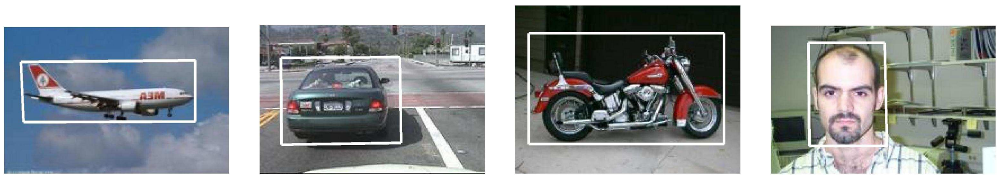


Figure 5.3.: Examples of annotated images from the Caltech4 dataset.

word. So, even for a small collection of pixels, the extraction of descriptor histograms is a computationally expensive job.

Another issue of the visual vocabulary framework, is the fact that images have different dimensions, thus a variable amount of response vectors per object. While small variations in the number of response vectors can be overcome through normalization of the descriptor histogram, very large variations can not be accounted for. Besides the fact that large variations in image dimensions contribute to noise, images which have high dimensions contribute to computational effort.

5.2.1. Computation Reduction through Grid Sampling

One way to reduce the computational effort is to limit the amount of image descriptors we use by utilizing only a subset of the image descriptors of an image. Where we would normally determine the response vectors of all pixels within the region of an object, now we only sample within a certain grid, thus reducing the amount of convolution operations that are needed. A reduced amount of response vectors would in turn reduce the amount of vectors that need to be mapped to an image descriptor with use of the visual vocabulary. Using grid sampling would be a theoretically sound solution, due to the fact that we do not expect the response vectors to differ rigorously within a local region. To some extend the histogram bins of the descriptor histogram will be proportionally reduced when using grid sampling.

5.2.2. Efficient Nearest Neighbour Search Using KD-trees

The assignment of response vectors to visual words is done by nearest neighbour search. A straight forward method for this assignment is exhaustive search. The distance of a response vector to all the vectors within the visual vocabulary is calculated and we assign a response vector to the visual word that lies closest. The complexity for this exhaustive search, given a visual vocabulary of n words, is $O(n)$. Considering we need to find the nearest neighbour for all response vectors, the complexity becomes $O(rn)$, given a collection of r response vectors. The complexity of exhaustive search confirms the computational effort of the visual vocabulary framework. However, more efficient manners in finding the nearest neighbour exist, which could improve the efficiency of our approach.

The k-dimensional binary search tree, or kd-tree for short, is a data structure that is really efficient for calculating the nearest neighbour [29, 30]. It is a space partitioning data structure that can organizes points in an k dimensional space, by using hyperplanes placed perpendicular to the coordinate axes. The partitions are arranged in a hierarchical manner to form a tree.

A kd-tree is formed as follows. First one of the points in your dataset (in this case our visual vocabulary) is chosen as the root node. Usually this is done by selecting a coordinate axis and choosing the median point to be our root node. The root node is used to form the first splitting hyperplane, by placing a plane perpendicular to the currently selected axis. Points are added by placing them in either of the two child nodes, depending on which side of the hyperplane passing through the root node they are situated. Subsequently, another axis is selected to act as the next splitting axis, after which a new median point is selected and the whole procedure is repeated. This process is recursively applied on the left and right children, until a small number of points remain. As a result of the hierarchical separation by hyperplanes, the space is partitioned into several hyper-rectangular regions called buckets, each containing several points.

The kd-tree can be searched for nearest neighbours by following a simple procedure. The coordinates of a new point are used to descend the tree and find the bucket that contains it, after which exhaustive search is performed to determine the closest point within that bucket. Since the exhaustive search is applied on only a very small number of points, and traversing the tree requires only a small number of operations, this is a significant improvement to the exhaustive search approach.

The speedup is also illustrated by the complexity of the kd-tree approach. Given a visual vocabulary of n words, the construction of a kd-tree has a theoretical complexity of $O(n \log_2 n)$. The expected complexity of a nearest neighbour search, given a collection of r response vectors, is $O(r \log_2 n)$, with a worst case complexity of $O(rn)$. Theoretically, the use of kd-trees will improve the overall efficiency of the visual vocabulary framework.

5.2.3. Reduction of Image Dimensions

The reduction of image dimensions will reduce the amount of pixels contained by the bounding boxes of each object. If we normalize the dimensions so that we decrease the overall dimensions of the images, we decrease the overall amount of pixels. By decreasing the amount of pixels, we will decrease the amount of computational effort, due to less convolution and mapping operations, so it could be used to improve efficiency.

However, if we use normalization of image dimensions to reduce the image dimensions, we are dealing with lossy compression. This implies that we are reducing the amount of information about an object. Some information is lost, so we can expect that this will lead to an overall performance loss. On the other hand, normalizing image dimensions will also normalize the dimensions of depicted objects, thus reduce the variability throughout the objects of each category. This will lead to an overall increase of performance.

5.3. Vocabulary Construction

In this section we will discuss the details concerning the vocabulary construction process. As was mentioned in section 2.3, several implementation choices have to be made. These design choices include properties such as size of the vocabulary, clustering methods, and early or late fusion of filter responses.

The clustering method that we applied is an implementation of the k-means clustering algorithm by D. Pelleg and A. More [31]. As was stated before, the results of k-means clustering depend heavily on the initialization. It is not guaranteed that k-means clustering will locate the optimal clustering, i.e. the clustering that minimizes the total within-group sum of squares distance (equation (2.1)). For this reason clustering was performed multiple times with different initializations. Each clustering experiment was repeated 5 times for 200 iterations, where each repetition made use of a different random initialization. Subsequently, the partitioning with the lowest total within-group sum of squares distance was selected.

We experimented with two different approaches in the vocabulary building process, early and late fusion of filter responses. Early fusion performs a single clustering to partition the data. Late fusion however, performs multiple clustering processes on subsets of the data, which are then fused. The number of clustering processes that late fusion performs is equal to the amount of classes of the dataset, because for each class a separate vocabulary is constructed. In order to be able to compare early and late fusion experiments, the final vocabulary sizes for both fusion approach should be equal.

The parameters used for the experiments are presented below.

fusion method	:	early
number of vocabularies	:	1
vocabulary sizes	:	$\{100, 250, 500, 1000, 1500, 2000, 2500, 3000\}$
fusion method	:	late
number of vocabularies	:	4
vocabulary sizes	:	$\{25, 65, 125, 250, 375, 500, 625, 750\}$

In the case of late fusion, the final vocabulary size is determined by the product of the number of vocabularies that will be fused and the size of the individual vocabularies. So, all vocabulary sizes should be multiplied by 4.

To sum up, a total of 16 vocabularies were constructed. In the case of early fusion, the experiments were repeated 5 times for each of the vocabulary sizes, giving a total of 40 experiments. With late fusion, the experiments were repeated 5 times for each of the sub-vocabularies and for each of the vocabulary sizes, giving a total of 160 experiments. Overall, a total of 200 experiments were conducted to construct 16 distinct vocabularies.

5.3.1. Filter Banks

Due to the extreme computational effort of the experiments we decided that we would only use one filter bank. The only filter bank we used in the experiments is the Winn-Criminisi-Minka filter bank [7], introduced in section 2.2, because it was the only filter bank that was specifically optimized and tested on object classification. All other filter banks discussed in section this were designed for texture classification, thus probably less suitable for object classification.

5.4. Classification Results

The following sections present the classification results. Some general evaluation approaches are addressed, and the final results of the various classification methods are discussed.

5.4.1. SVM Parameter Search

One of the disadvantages of the SVM classification approach is that SVM requires some input parameters. For each parameter setting the classification results will be distinct, and can differ considerably in performance. Beforehand it is not known which parameters are best suited for the classification problem. For this reason we need to find a way to determine the parameters for the SVM that result in the highest performance. Since no efficient approach to parameter search exists for SVM, the only way to determine the correct parameter settings is by grid search. With grid search multiple parameter setting are evaluated, using a grid of different parameter values.

Since we only used the RBF kernel in our experiments, we will only discuss the different parameters that can be applied on this kernel. For the RBF kernel only two parameters are needed, the parameter C, and the γ in equation 5.1. The parameter C specifies the cost of a misclassification within the training data, thus describes the trade-off between training error and margin. The parameter γ is used in the kernel and determines the RBF radius.

$$K(u,v) = \exp\left(-\gamma * |u-v|^2\right) \tag{5.1}$$

Several parameter values for the C and γ parameters of the RBF kernel were applied. For both the parameters of the RBF kernel 11 parameter values are applied, thus creating 121 different pairs of parameters. For each of the 121 distinct parameter settings experiments were conducted to evaluate the performance. The parameter values that we used in the experiments are the following.

$$\begin{aligned} C &: \{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100\} \\ \gamma &: \{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100\} \end{aligned}$$

The evaluation of the parameter settings is based on 5-fold cross validation to prevent overfitting. Overfitting can occur when the performance evaluation is solely based on the training error. This can cause the learned classifier to conform to the training data, the risk being that the classifier is too specific. The classifier might be perfect for the training set, while on new instances the model performs poorly. Using cross validation, we can prevent overfitting.

When applying cross validation, we use the test error to evaluate the parameters. The data is first divided into 5 equally sized subsets. Subsequently, each set is tested using the classifier trained using the remaining subsets. So, for each parameter setting 5 separate evaluations are performed, which are combined to form the full evaluation.

5.4.2. SVM Results

The SVMs are evaluated for each of the different parameter settings mentioned in the previous section. We evaluate the SVM classifiers by the overall accuracy of classification. This means that for each of the parameter settings, an individual object class model is learned, and evaluated in combination with the other object class models. In a sense we measure the combined accuracy of the object class models. The optimal accuracy for the two different fusion approaches, and the various vocabulary sizes are shown in figure 5.4. Appendix A shows the grid results for each of the parameter settings, the different vocabulary sizes, and the various fusion approaches.

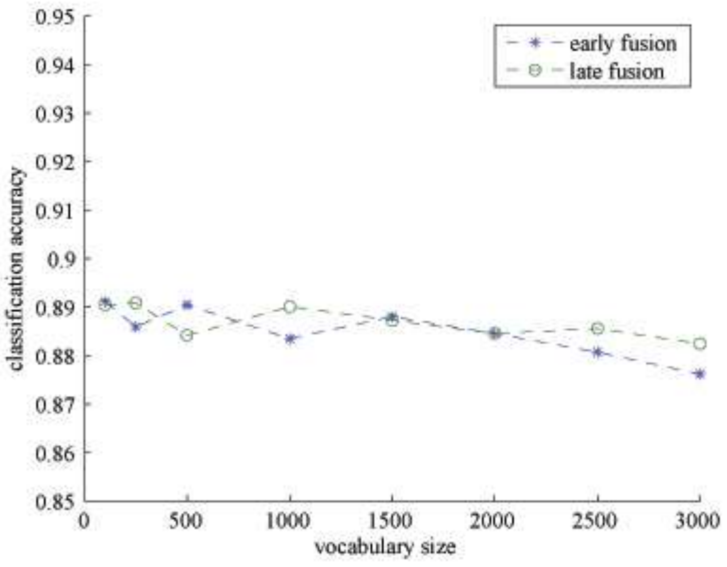


Figure 5.4.: Classification accuracy for the SVM classifiers. For each vocabulary size, only the optimal classification accuracy is plotted.

The figure shows that most of the different classifiers perform equally well. The general classification accuracy is between 88% and 89%. Furthermore, it seems that vocabularies constructed by late fusion generally have a higher accuracy than those constructed by early fusion. What is more, an increasing vocabulary size appears to have a slight negative effect on the classification accuracy. This could be an effect of data sparseness, which increases when we increase the vocabulary size due to the fact that we have an equal amount of response vectors, but a larger amount of histogram bins.

5.4.3. Naive Bayes Results

As with the SVM classification results, the evaluation for both the naive Bayes and the simplified naive Bayes model is based on 5-fold cross validation. The total dataset is divided into 5 subsets, of which 4 are used for training and 1 is used for validation. In the case of the naive Bayes model, we determine both mean and variance from the training set. For the simplified version, only the mean of each histogram bin is determined, and we assume a constant variance. Experiments were performed for both early and late fusion with 8 different vocabulary sizes. Figure 5.5 shows the results for the different Bayesian models.

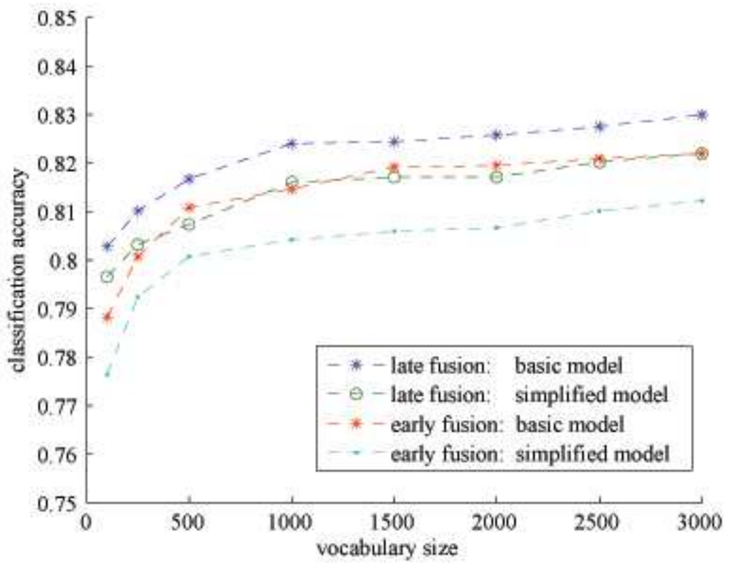


Figure 5.5.: Classification accuracy for both naive Bayes classifiers.

Looking at the overall performance of the Bayesian models, it is evident that all models profit from larger vocabularies. Especially within the segment with smaller vocabulary sizes, the differences are significant. Between the performances of the larger vocabulary sizes, the differences are less prominent. However, in general we observe an increase in accuracy with an increase of vocabulary size.

Furthermore, if we look at the difference between the simplified and naive Bayes model, we can see that the naive Bayes model always outperforms the simplified model. Additionally, it is save to say that the models based on the vocabularies constructed by late fusion have a higher accuracy than those based on the early fusion approach. This is particularly illustrated by the fact that the simplified naive Bayes model based on late fusion has a similar performance as the naive Bayes model based on late fusion.

If we compare these results to the results of the SVM classifiers in figure 5.4, we notice that the SVMs have a significantly higher accuracy than the naive Bayes classifiers. The difference between the SVM classifiers and the naive Bayes classifiers is around 7% on average.

5.5. Localization Results

The performance of the localization task is evaluated by comparing the predicted region B_p, detected by the localization algorithm, with the true region B_t, provided by manual annotation. We calculate precision-recall curves as described in the Pascal VOC challenge[1]. First we determine the area of overlap by:

$$a = \frac{\text{area}\left(B_p \bigcap B_t\right)}{\text{area}\left(B_p \bigcup B_t\right)} \tag{5.2}$$

If $a > 0.5$, B_p is considered a correct prediction, otherwise it is considered a false prediction. The confidence scores corresponding to the predicted regions, which are returned by the localization algorithm, are used to compute the precision-recall curves. By applying a set of thresholds to the confidence output that corresponds to the various detections, we can plot a precision-recall curve. Recall is the proportion of the object instances in the dataset that has been detected, and precision is the percentage of detections that are correct. More formally, recall is defined by TP/nP and precision by TP/(TP+FP), where TP are the number of true positives, FP the number of false positives, and nP the total number of object instances in the complete dataset.

Another measure which we will use to compare different localization algorithms, is the average precision. Basically, average precision is the area under the precision-recall curve, and indicates the average performance of the localization algorithms. Average precision is used in order to quickly compare different localization algorithms. However, the precision-recall curve can show particular results, that can not be summarized by a single value, and this is why we use both precision-recall curves, and average precision.

5.6. Tile-Based Approach

As was discussed in section 4.1, the tile-based approach to object localization is related to the classification task. The classifiers that are learned during the classification task, can be directly applied in the detection task, using the tile-based approach. Throughout the experiments, the classifier are used to evaluate several selected regions, giving each region a confidence score (the class conditional probability), indicating the probability that an object is present within the evaluated region.

Since it is impractical to evaluate each possible region within an image (there might be millions), a simple solution is applied. The selection procedure for the 'to-be-evaluated' regions is as follows. For each object class, the average width and height of the object region is determined, as well as the standard deviation. The size and shape of the 'tile' is determined by the average and standard deviation. Five different scales of the mean object region are used to evaluate an image on various positions. The scales are determined by the standard deviation. In other words, five different tiles are used.

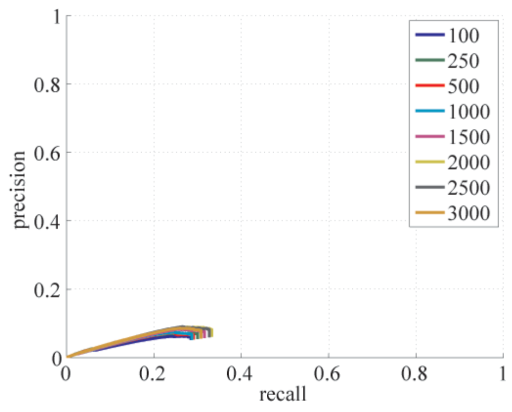
Each of the classification methods that were applied in the recognition experiments are reused for object detection. In the case of the SVM classifier, only one parameter

[1]http://www.pascal-network.org/challenges/VOC/

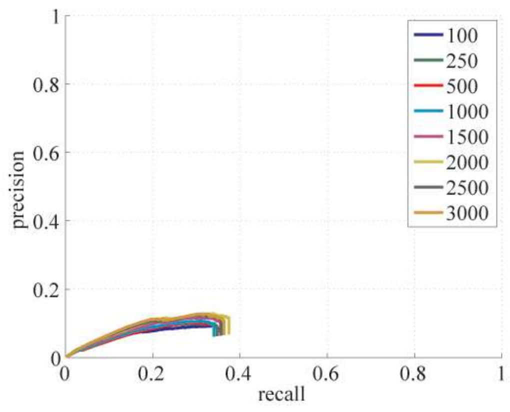
setting is used. The parameter setting is selected by looking at the optimal classification performance.

5.6.1. Simplified Naive Bayes Results

Figures 5.6 through 5.9 illustrate the results of the simplified naive Bayes method for all the different object classes using both early and late fusion. Tables 5.1 through 5.4 contain the average precision for the different localization experiments using the simplified naive Bayes method. For each class there is a separate figure and corresponding table.



(a) early fusion



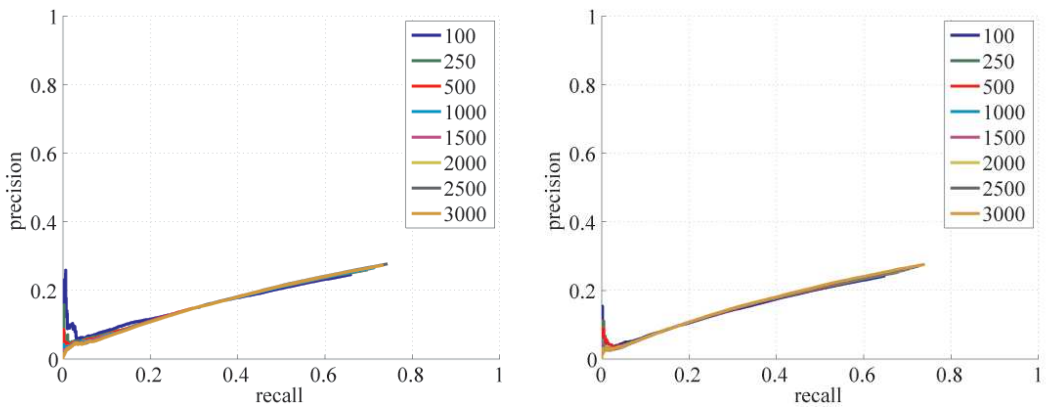
(b) late fusion

Figure 5.6.: Localization results of the cars class, using the simplified naive Bayes model.

	Average Precision	
Model	early	late
simplified naive Bayes: voc size 100	0.018	0.035
simplified naive Bayes: voc size 250	0.024	0.034
simplified naive Bayes: voc size 500	0.021	0.034
simplified naive Bayes: voc size 1000	0.021	0.036
simplified naive Bayes: voc size 1500	0.028	0.044
simplified naive Bayes: voc size 2000	0.030	0.049*
simplified naive Bayes: voc size 2500	0.030**	0.047
simplified naive Bayes: voc size 3000	0.029	0.048

Table 5.1.: Average precision of the cars class, using the simplified naive Bayes model.

The performance of the simplified naive Bayes model for the cars class, illustrated in figure 5.6 and table 5.1, show very poor results. For each of the vocabulary sizes, the average precision is around 3% for early fusion and around 5% at best for late fusion. This indicates that on average only 3-5 out of a hundred detections is correct. What is more, the optimal recall value is around 3.8%, which tells us that very few objects in the dataset are actually detected. Moreover, the shape of the precision-recall curve is not what we would expect. When the confidence threshold is lowered, in order to construct the precision-recall curve, we notice an increase of precision as well as an increase in recall. This implies that the detections with the highest confidence are incorrect. Furthermore, for both fusion approaches it is clear that larger vocabularies result in higher average precision.



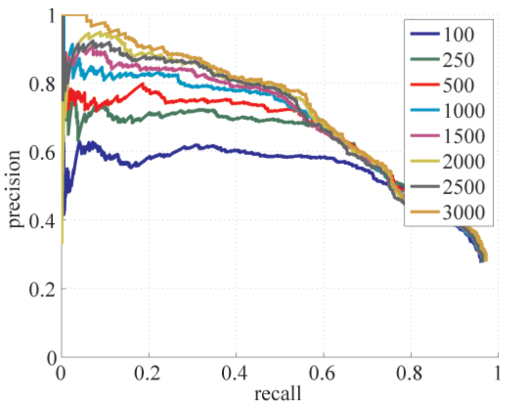
(a) early fusion (b) late fusion

Figure 5.7.: Localization results of the airplanes class, using the simplified naive Bayes model.

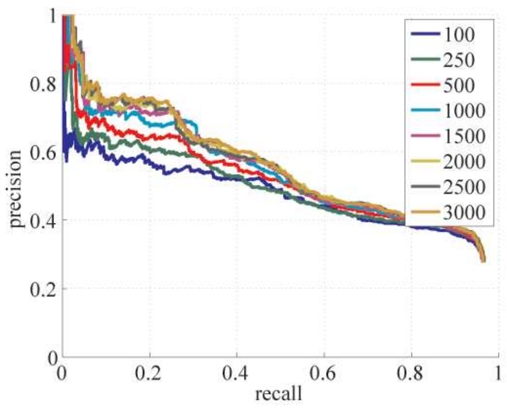
	Average Precision	
Model	early	late
simplified naive Bayes: voc size 100	0.165	0.150
simplified naive Bayes: voc size 250	0.174	0.169
simplified naive Bayes: voc size 500	0.189	0.173
simplified naive Bayes: voc size 1000	0.190	0.188
simplified naive Bayes: voc size 1500	0.195	0.191
simplified naive Bayes: voc size 2000	0.195	0.195
simplified naive Bayes: voc size 2500	0.198**	0.193
simplified naive Bayes: voc size 3000	0.196	0.197*

Table 5.2.: Average precision of the airplanes class, using the simplified naive Bayes model.

The performance of the simplified naive Bayes model for the airplanes class, illustrated in figure 5.7 and table 5.2, is reasonable. For each of the vocabulary sizes, the optimal average precision is around 20% for both fusion approaches. Actually, the early fusion approach performs a bit better than the late fusion approach. However, this appears to be consequence of the relatively high precision at the beginning of the precision-recall curve. The illustrations also show that the optimal recall value is 77%, indicating that almost 4 out of 5 objects are detected. Like with the cars class, the shape of the precision-recall curve is not what we would expect, since the slope of the curve is positive. We notice an increase of precision, when the confidence threshold is lowered. This implies that the detections with the highest confidence are incorrect. Furthermore, larger vocabularies result in higher average precision.



(a) early fusion



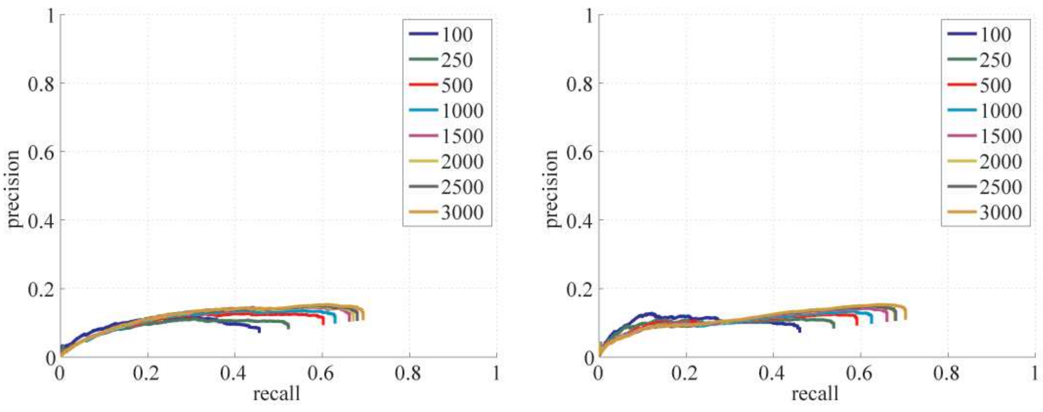
(b) late fusion

Figure 5.8.: Localization results of the motorbikes class, using the simplified naive Bayes model.

	Average Precision	
Model	early	late
simplified naive Bayes: voc size 100	0.548	0.482
simplified naive Bayes: voc size 250	0.614	0.495
simplified naive Bayes: voc size 500	0.637	0.517
simplified naive Bayes: voc size 1000	0.654	0.536
simplified naive Bayes: voc size 1500	0.666	0.544
simplified naive Bayes: voc size 2000	0.687	0.551
simplified naive Bayes: voc size 2500	0.675	0.550
simplified naive Bayes: voc size 3000	0.700**	0.554*

Table 5.3.: Average precision of the motorbikes class, using the simplified naive Bayes model.

Figure 5.8 and table 5.3, illustrate the performance of the simplified naive Bayes model for the motorbikes class. The optimal average precision is 70% for the early fusion approach, and around 55% for late fusion. This means that early fusion outperforms the late fusion approach. The precision-recall curves are like we would expect and a recall value of around 98% shows that nearly every object is detected. Again, larger vocabularies result in higher average precision.



(a) early fusion (b) late fusion

Figure 5.9.: Localization results of the faces class, using the simplified naive Bayes model.

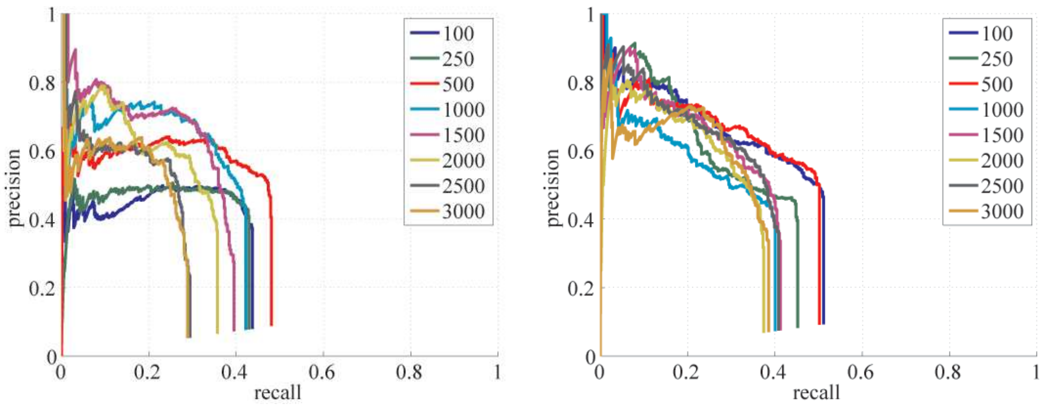
	Average Precision	
Model	early	late
simplified naive Bayes: voc size 100	0.053	0.054
simplified naive Bayes: voc size 250	0.058	0.058
simplified naive Bayes: voc size 500	0.078	0.071
simplified naive Bayes: voc size 1000	0.084	0.083
simplified naive Bayes: voc size 1500	0.097	0.095
simplified naive Bayes: voc size 2000	0.099	0.099
simplified naive Bayes: voc size 2500	0.100	0.099
simplified naive Bayes: voc size 3000	0.102**	0.109*

Table 5.4.: Average precision of the faces class, using the simplified naive Bayes model.

The performance of the simplified naive Bayes models for the faces class, illustrated in figure 5.9 and table 5.4, is poor. For each of the vocabulary sizes, the optimal average precision is around 11%. However, the optimal recall value shows that around 70% of the objects are detected. The results for the different fusion approaches are very similar. Like with the cars and the airplane class, we see an increase of precision, when the confidence threshold is lowered. This implies that the detections with the highest confidence are incorrect. Once more, a higher average precision is obtained with the use of larger vocabularies.

5.6.2. Naive Bayes Results

Figures 5.10 through 5.13 illustrate the results of the tile-based approach using the naive Bayes model. Tables 5.5 through 5.8 contain the corresponding average precision for the different localization experiments using the naive Bayes model. For each class and fusion approach there is a separate figure and corresponding table.



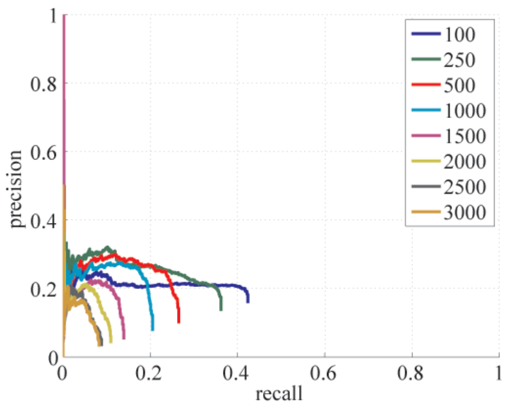
(a) early fusion (b) late fusion

Figure 5.10.: Localization results of the cars class, using the naive Bayes model.

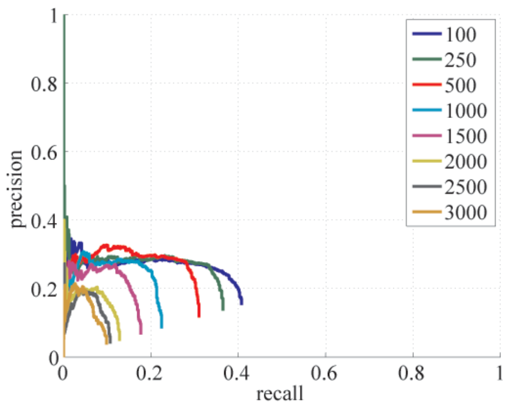
Model	Average Precision	
	early	late
naive Bayes: voc size 100	0.214	0.370*
naive Bayes: voc size 250	0.214	0.319
naive Bayes: voc size 500	0.300	0.370*
naive Bayes: voc size 1000	0.309**	0.266
naive Bayes: voc size 1500	0.286	0.305
naive Bayes: voc size 2000	0.259	0.261
naive Bayes: voc size 2500	0.192	0.305
naive Bayes: voc size 3000	0.191	0.263

Table 5.5.: Average precision of the cars class, using the naive Bayes model.

Figure 5.10 and table 5.5 show the results for the cars class using the naive Bayes model. We can see that the model using late fusion performs a slightly better. Also, the shape of the precision-recall curve is like we expected. In contrast with to the simplified naive Bayes model, it seems that smaller dictionaries perform better than larger ones. Although the smaller vocabularies do not seem to have the highest precision, the increased recall of the smaller vocabularies makes all the difference. This could indicate that a higher expressiveness is, to some degree, beneficial for the accuracy of the detection, while being less good in detecting every single object. Furthermore, we can see a significant improvement with respect to the simplified naive Bayes model.



(a) early fusion



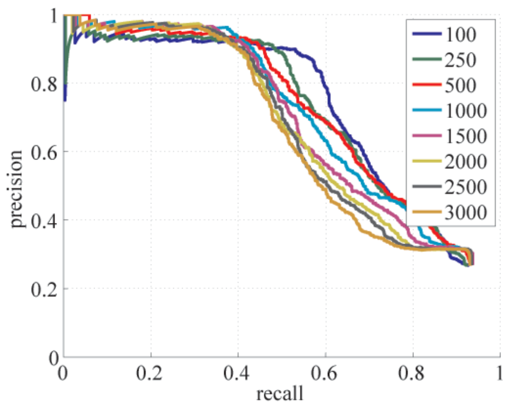
(b) late fusion

Figure 5.11.: Localization results of the airplanes class, using the naive Bayes model.

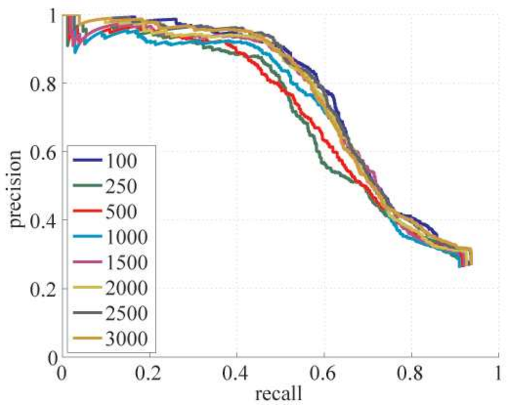
	Average Precision	
Model	early	late
naive Bayes: voc size 100	0.132**	0.127
naive Bayes: voc size 250	0.104	0.142*
naive Bayes: voc size 500	0.093	0.101
naive Bayes: voc size 1000	0.094	0.070
naive Bayes: voc size 1500	0.069	0.049
naive Bayes: voc size 2000	0.036	0.037
naive Bayes: voc size 2500	0.024	0.027
naive Bayes: voc size 3000	0.031	0.019

Table 5.6.: Average precision of the airplanes class, using the naive Bayes model.

Figure 5.11 and table 5.6 show the results for the airplanes class using the naive Bayes model. Again, the smaller vocabularies result in a higher average precision than the larger vocabularies. However, when we look at the precision of the vocabularies, the early fusion approach in particular, we notice that some larger vocabularies perform better, although the recall is lower. This is exactly the same observation that we made with the previous object class. Furthermore, consistent with all the previous results, the late fusion approach has a higher average precision.



(a) early fusion



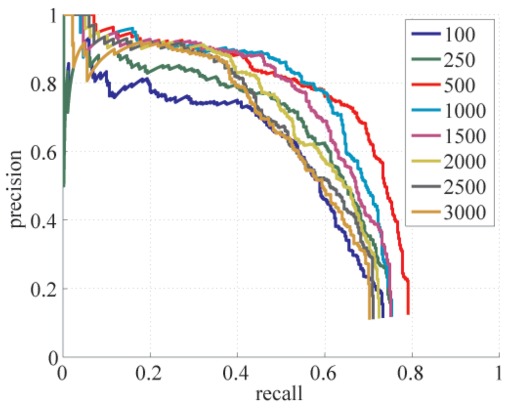
(b) late fusion

Figure 5.12.: Localization results of the motorbikes class, using the naive Bayes model.

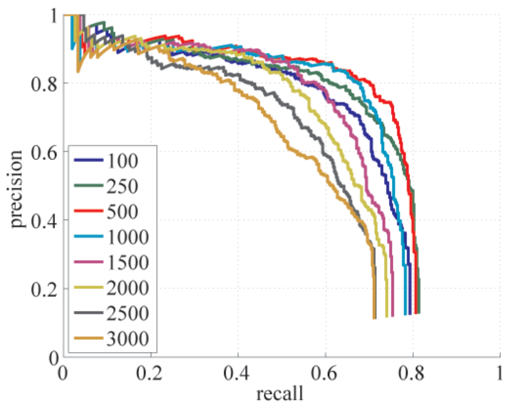
	Average Precision	
Model	early	late
naive Bayes: voc size 100	0.701**	0.714*
naive Bayes: voc size 250	0.693	0.659
naive Bayes: voc size 500	0.692	0.666
naive Bayes: voc size 1000	0.679	0.679
naive Bayes: voc size 1500	0.661	0.696
naive Bayes: voc size 2000	0.647	0.699
naive Bayes: voc size 2500	0.643	0.712
naive Bayes: voc size 3000	0.631	0.705

Table 5.7.: Average precision of the motorbikes class, using the naive Bayes model.

Figure 5.12 and table 5.7 show the results for the motorbikes class using the naive Bayes model. Once more, smaller vocabularies have a higher average precision than the larger vocabularies. However, as we saw earlier, if we look at the precision recall curves of the various vocabulary sizes, we see that the larger vocabularies have a higher precision, while the smaller ones have a higher average precision. Once more, the results show that late fusion has a higher performance. Also, we see an improvement of performance with respect to the simplified naive Bayes model.



(a) early fusion



(b) late fusion

Figure 5.13.: Localization results of the faces class, using the naive Bayes model.

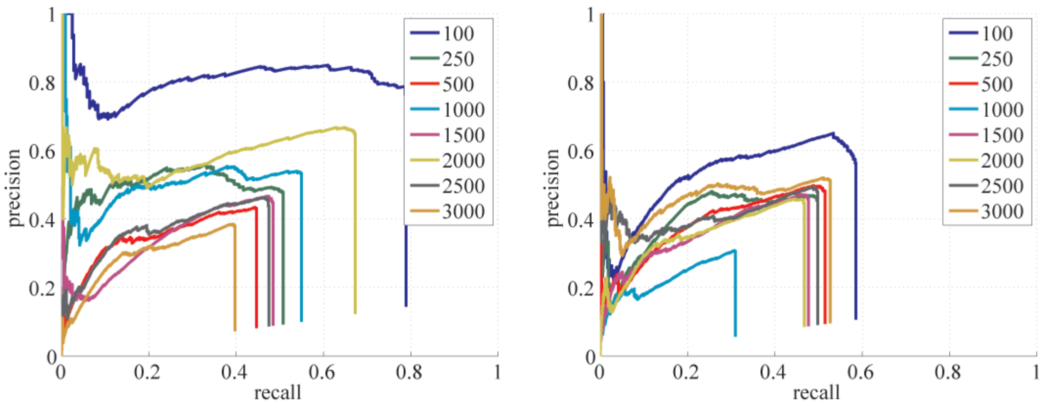
	Average Precision	
Model	early	late
naive Bayes: voc size 100	0.493	0.637
naive Bayes: voc size 250	0.554	0.682
naive Bayes: voc size 500	0.642**	0.699*
naive Bayes: voc size 1000	0.620	0.662
naive Bayes: voc size 1500	0.605	0.626
naive Bayes: voc size 2000	0.566	0.593
naive Bayes: voc size 2500	0.547	0.555
naive Bayes: voc size 3000	0.532	0.537

Table 5.8.: Average precision of the faces class, using the naive Bayes model.

Figure 5.13 and table 5.8 show the results for the faces class using the naive Bayes model. Once more, smaller vocabularies have a higher average precision than the larger vocabularies. Additionally, a similar trend as we noticed earlier is visible in the precision-recall curves. Larger vocabularies have a higher precision, and a lower recall and average precision. Again, the late fusion approach has a higher performance than the early fusion approach.

5.6.3. SVM Results

Figures 5.14 through 5.17 illustrate the results of the tile-based approach in combination with the SVM classifiers. Tables 5.9 through 5.12 contain the corresponding average precision for the different localization experiments. For each class and fusion approach there is a separate figure and corresponding table.



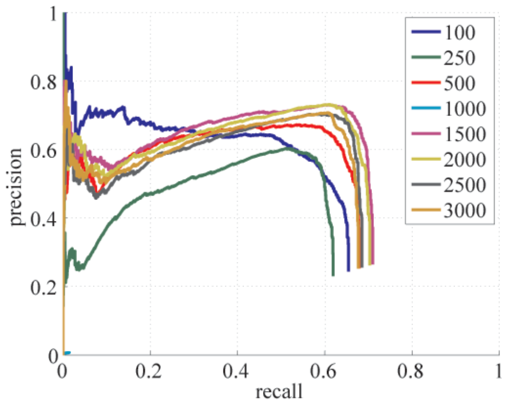
(a) early fusion (b) late fusion

Figure 5.14.: Localization results of the cars class, using the SVM classifiers.

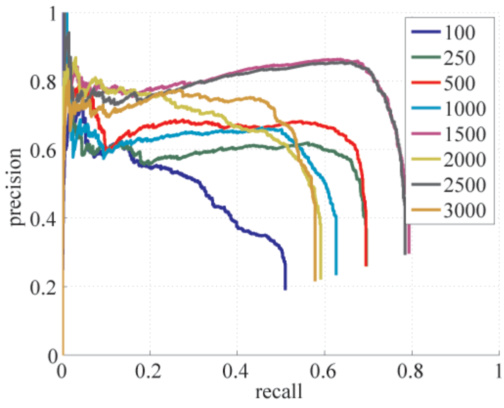
Model	Average Precision	
	early	late
SVM: voc size 100	0.650**	0.387*
SVM: voc size 250	0.282	0.227
SVM: voc size 500	0.186	0.285
SVM: voc size 1000	0.310	0.103
SVM: voc size 1500	0.248	0.225
SVM: voc size 2000	0.461	0.219
SVM: voc size 2500	0.220	0.259
SVM: voc size 3000	0.147	0.295

Table 5.9.: Average precision of the cars class, using the SVM classifiers.

Figure 5.14 and table 5.9 show the results for the cars class using the tile-based approach and the SVM classifiers. For both fusion methods, the smallest vocabulary shows the optimal results. However, it seems that the performance is not dependent on vocabulary size. For both fusion approaches, the two highest scoring vocabularies are either very small or very large. Furthermore, it is hard to say whether the early fusion or late fusion approach has a higher overall performance, but obviously the optimal performance is achieved by early fusion.



(a) early fusion



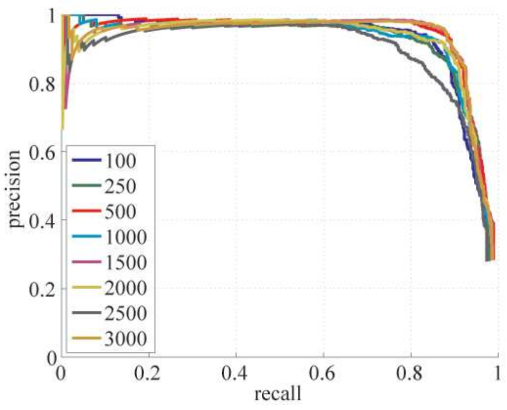
(b) late fusion

Figure 5.15.: Localization results of the airplanes class, using the SVM classifiers.

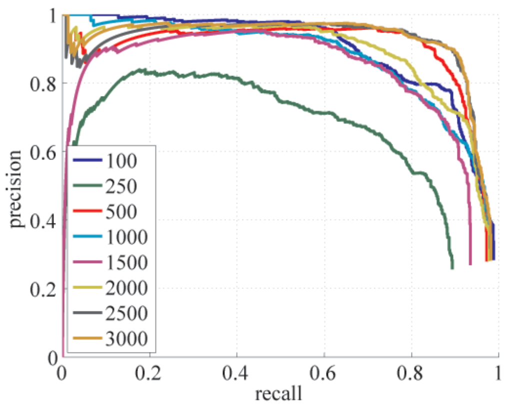
	Average Precision	
Model	early	late
SVM: voc size 100	0.438	0.275
SVM: voc size 250	0.386	0.411
SVM: voc size 500	0.442	0.472
SVM: voc size 1000	0.000	0.416
SVM: voc size 1500	0.515**	0.648
SVM: voc size 2000	0.505	0.412
SVM: voc size 2500	0.468	0.650*
SVM: voc size 3000	0.472	0.425

Table 5.10.: Average precision of the airplanes class, using the SVM classifiers.

Figure 5.15 and table 5.10 show the results for the airplanes class using the tile-based approach and the SVM classifiers. The precision-recall curves of the early fusion experiments show similar results for each vocabulary size. Only the two smallest vocabularies vary from the main trend. The results of the late fusion experiments also seem fairly similar, except for a vocabulary size of 1500 or 2500. Furthermore, the late fusion approach performs better than the early fusion approach.



(a) early fusion



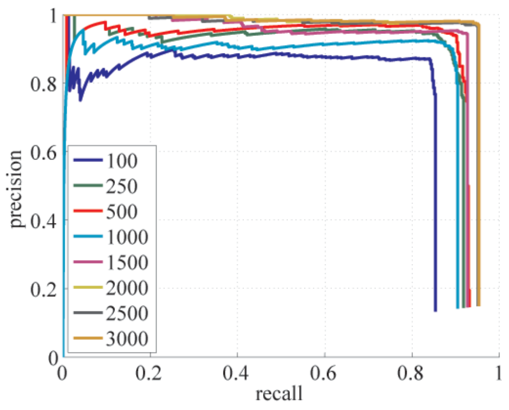
(b) late fusion

Figure 5.16.: Localization results of the motorbikes class, using the SVM classifiers.

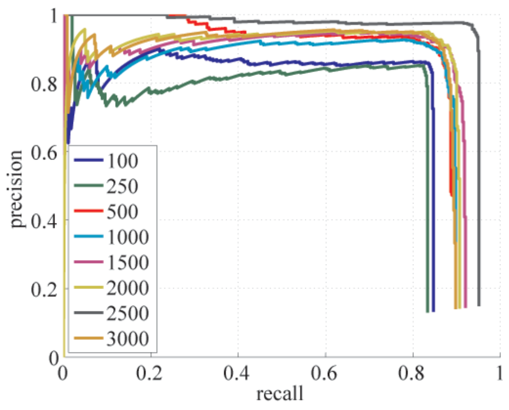
	Average Precision	
Model	early	late
SVM: voc size 100	0.900	0.872
SVM: voc size 250	0.899	0.642
SVM: voc size 500	0.915**	0.887
SVM: voc size 1000	0.904	0.851
SVM: voc size 1500	0.915**	0.812
SVM: voc size 2000	0.906	0.865
SVM: voc size 2500	0.879	0.899
SVM: voc size 3000	0.912	0.902*

Table 5.11.: Average precision of the motorbikes class, using the SVM classifiers.

Figure 5.16 and table 5.11 show the results for the motorbikes class using the tile-based approach and the SVM classifiers. The results are significantly better than those of the generative models. The results show that on average the early fusion approach performs better than the late fusion approach. The vocabulary size does not seem to have an affect on the performance of the localization algorithms.



(a) early fusion



(b) late fusion

Figure 5.17.: Localization results of the faces class, using the SVM classifiers.

	Average Precision	
Model	early	late
SVM: voc size 100	0.760	0.715
SVM: voc size 250	0.863	0.696
SVM: voc size 500	0.878	0.824
SVM: voc size 1000	0.835	0.796
SVM: voc size 1500	0.880	0.851
SVM: voc size 2000	0.941**	0.855
SVM: voc size 2500	0.938	0.935*
SVM: voc size 3000	0.941**	0.819

Table 5.12.: Average precision of the faces class, using the SVM classifiers.

Figure 5.17 and table 5.12 show the results for the faces class using the tile-based approach and the SVM classifiers. In comparison to the generative models, the SVM classifiers have significantly higher performance. Furthermore, larger vocabularies result in a higher average precision. Also, the early fusion approach has a higher performance than the late fusion approach.

5.7. Histogram Tracking Approach

In this section we will present the results of the histogram tracking approach, discussed in section 4.2. As was mentioned, the histogram tracking approach is based region optimization using an adapted form of mean shift. In order to optimize regions with respect to their position and size, we need to use a reference object model. The simplified naive Bayes model is used as reference model.

The histogram tracking approach uses a number of starting regions, which are later optimized. This means that we need to specify the number shape and size of the starting regions. We have experimented with several numbers and sizes of starting regions. Figure 5.18 and 5.19 show some examples of localization using histogram tracking. In figure

5.18 we used a single starting regions, which spans the entire image. Figure 5.19 shows an example were we used 25 starting regions as input for the optimization algorithm.

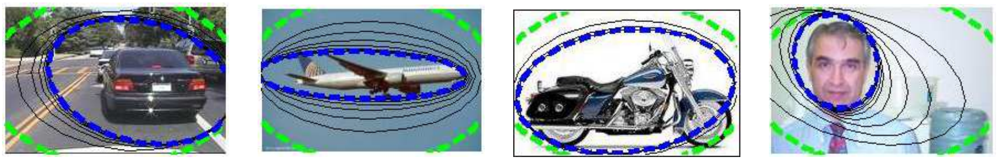


Figure 5.18.: Examples of localization using the histogram tracking approach with a single starting region. The green ellipse represents the starting region, the black ellipses the intermediate update steps, and the blue ellipse the final detection.

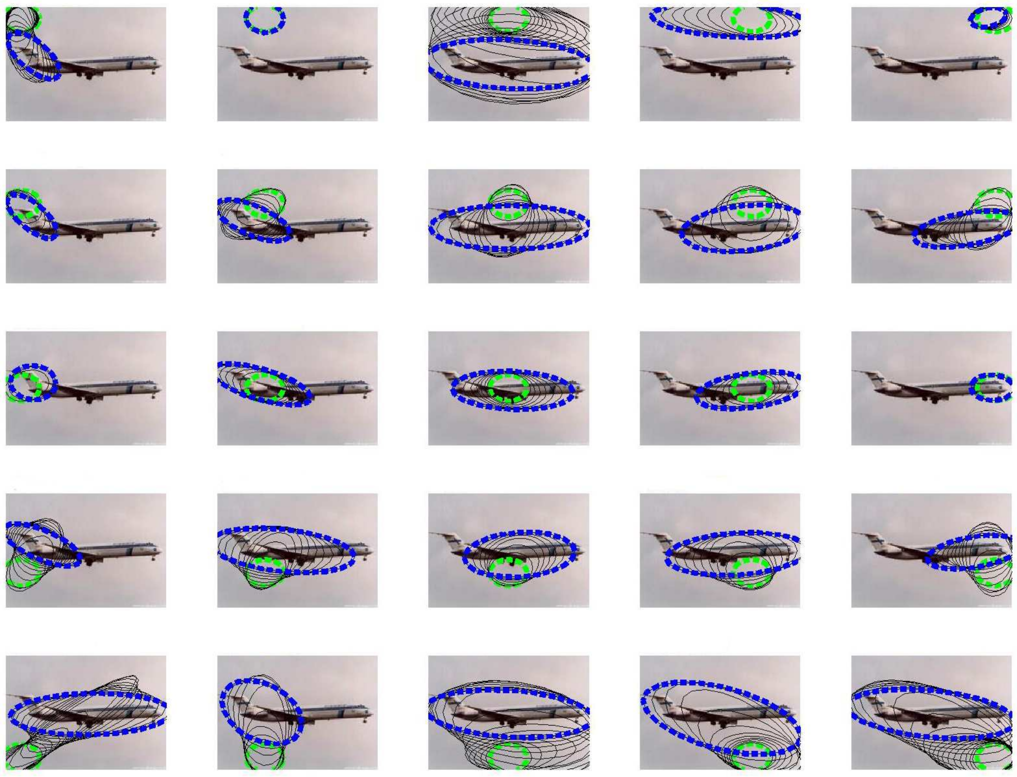
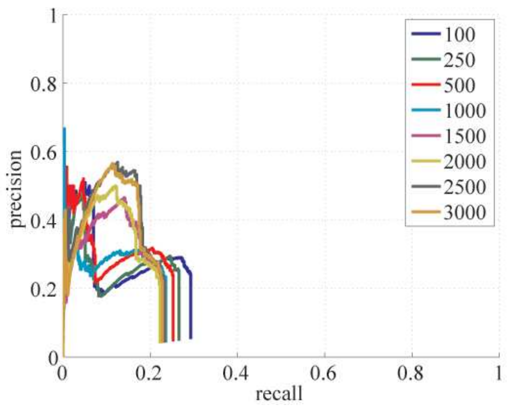


Figure 5.19.: Example of localization using the histogram tracking approach with 25 starting regions. The green ellipses represents the starting regions, the black ellipses the intermediate update steps, and the blue ellipses the final optimized region.

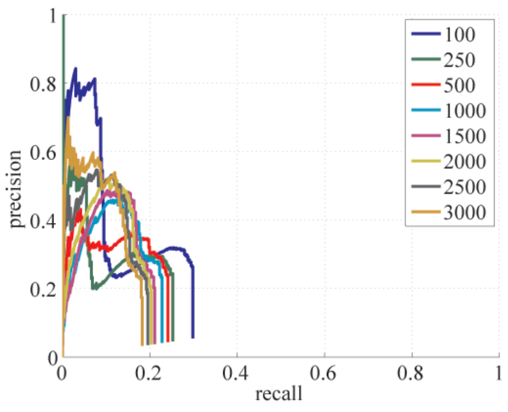
We have conducted several experiments with use of the histogram tracking approach. Again, we have tested the approach with several vocabularies, varying in size and fusion method. The same vocabularies that were used in the experiments of the tile-based approach, are applied in these experiments. In total we conducted localization experiments for 16 different vocabularies. The results shown in the next sections are based on localization using a single starting region, since it yielded in the highest performance.

5.7.1. Histogram Tracking Results

Figures 5.20 through 5.23 illustrate the results of the histogram tracking approach. Tables 5.13 through 5.16 contain the average precision for the different localization experiments using histogram tracking. For each class there is a separate figure and corresponding table.



(a) early fusion



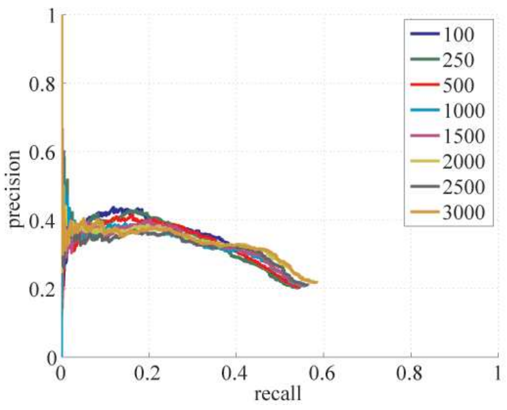
(b) late fusion

Figure 5.20.: Localization results of the cars class, using the histogram tracking approach.

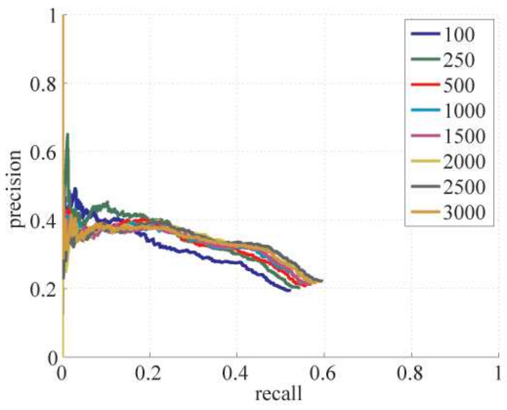
	Average Precision	
Model	early	late
hisgtoram tracking: voc size 100	0.104	0.140*
hisgtoram tracking: voc size 250	0.097	0.127
hisgtoram tracking: voc size 500	0.106	0.087
hisgtoram tracking: voc size 1000	0.090	0.100
hisgtoram tracking: voc size 1500	0.099	0.104
hisgtoram tracking: voc size 2000	0.105	0.107
hisgtoram tracking: voc size 2500	0.121**	0.095
hisgtoram tracking: voc size 3000	0.119	0.101

Table 5.13.: Average precision of the cars class, using the histogram tracking approach.

The results of localization using the histogram tracking approach for the cars class is illustrated in figure 5.20 and table 5.13. The differences between the various vocabulary sizes are very insignificant and it is not clear whether small or large vocabularies are preferred. If we compare the results of this approach with the results of the tile-based approach in combination with the simplified naive Bayes model, we notice a significant increase of performance. However, when we compare it to the tile-based approach in combination with the other object models, the histogram tracking approach has a lower performance. Additionally, the recall of the histogram tracking approach is lower, which indicates that less objects are detected. Furthermore, the results show that late fusion tends to achieve a higher average precision than early fusion.



(a) early fusion



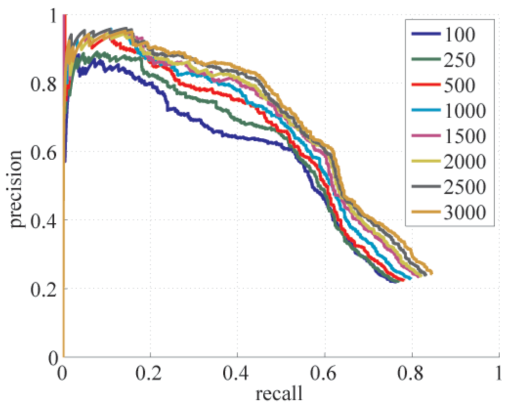
(b) late fusion

Figure 5.21.: Localization results of the airplanes class, using the histogram tracking approach.

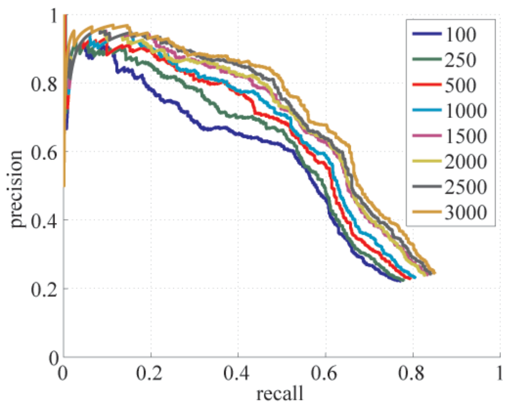
	Average Precision	
Model	early	late
hisgtoram tracking: voc size 100	0.193	0.175
hisgtoram tracking: voc size 250	0.198	0.201
hisgtoram tracking: voc size 500	0.194	0.200
hisgtoram tracking: voc size 1000	0.197	0.200
hisgtoram tracking: voc size 1500	0.225	0.201
hisgtoram tracking: voc size 2000	0.226**	0.209
hisgtoram tracking: voc size 2500	0.220	0.232*
hisgtoram tracking: voc size 3000	0.226**	0.226

Table 5.14.: Average precision of the airplanes class, using the histogram tracking approach.

Figure 5.21 and table 5.14 show the results for the airplanes class using the histogram tracking approach. The results for the various vocabulary sizes are very much alike. However, in contrast with the results of the tile-based approach, the results show that larger vocabularies perform better. Overall, the histogram tracking approach has a higher performance than the tile-based approach using both naive Bayes models. The SVM classifiers again, are superior to the histogram tracking approach. Once more, the late fusion approach yields higher average precision than the early fusion approach.



(a) early fusion



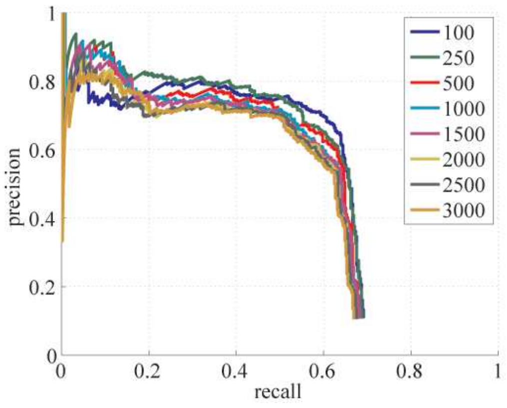
(b) late fusion

Figure 5.22.: Localization results of the motorbikes class, using the histogram tracking approach.

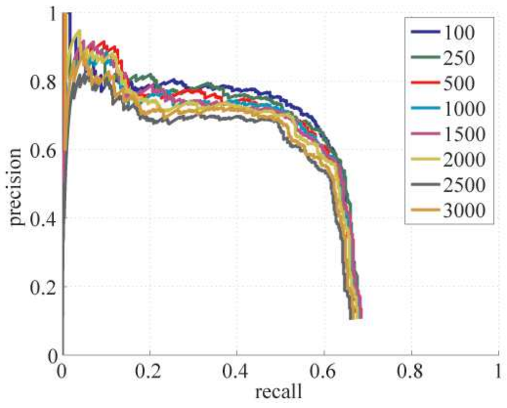
	Average Precision	
Model	early	late
hisgtoram tracking: voc size 100	0.487	0.491
hisgtoram tracking: voc size 250	0.509	0.518
hisgtoram tracking: voc size 500	0.531	0.545
hisgtoram tracking: voc size 1000	0.551	0.572
hisgtoram tracking: voc size 1500	0.577	0.598
hisgtoram tracking: voc size 2000	0.581	0.597
hisgtoram tracking: voc size 2500	0.592	0.612
hisgtoram tracking: voc size 3000	0.599**	0.631*

Table 5.15.: Average precision of the motorbikes class, using the histogram tracking approach.

Figure 5.22 and table 5.15 show the results for the motorbikes class using the histogram tracking approach. Equivalent to the previous object class, larger vocabularies results in higher precision and recall scores. If we compare the results to the tile-based approach in combination with the simplified naive Bayes model, we see a significant increase in performance. If we compare the results to the tile-based approach in combination with the other two object models, we see that the SVM model has a significantly higher performance, while the naive Bayes model shows similar results. Finally, the late fusion approach has a higher performance than the early fusion approach.



(a) early fusion



(b) late fusion

Figure 5.23.: Localization results of the faces class, using the histogram tracking approach.

	Average Precision	
Model	early	late
hisgtoram tracking: voc size 100	0.518	0.512
hisgtoram tracking: voc size 250	0.529**	0.513*
hisgtoram tracking: voc size 500	0.506	0.503
hisgtoram tracking: voc size 1000	0.498	0.496
hisgtoram tracking: voc size 1500	0.494	0.493
hisgtoram tracking: voc size 2000	0.481	0.487
hisgtoram tracking: voc size 2500	0.481	0.446
hisgtoram tracking: voc size 3000	0.475	0.471

Table 5.16.: Average precision of the faces class, using the histogram tracking approach.

Figure 5.23 and table 5.16 show the results for the faces class using the histogram tracking approach. The differences between the various vocabulary sizes is small. However, it seems that smaller vocabularies perform better than large vocabularies. Furthermore, the early fusion performs slightly better. If we compare the results to the tile-based approach, we see an increase of performance with respect to the simplified naive Bayes model. However, the tile-based approach shows a higher performance for both other object models.

5.8. Timing Results

An import aspect of localization approaches is their computational complexity. Often, this is a prominent drawback of many localization algorithms. We have timed the various localization experiments and we present the results here. The localization process is measured separately from the descriptor extraction process, so we only measure the time that the classifiers take to evaluate each image. Tables 5.17 show the amount of time each algorithm took to process the complete dataset.

	Processing Time							
	sim. naive Bayes		naive Bayes		SVM		hist. tracking	
voc. size	early	late	early	late	early	late	early	late
100	2:58:21	3:22:08	2:52:26	2:49:32	12:41:49	7:23:36	3:11:50	3:07:44
250	3:25:18	3:35:21	3:10:31	3:12:29	25:36:47	14:19:38	3:13:38	3:04:13
500	3:42:27	3:43:57	3:35:14	3:35:44	16:21:28	17:16:18	2:54:10	2:58:47
1000	4:10:01	4:16:33	4:30:27	4:35:12	29:44:02	33:39:04	2:43:28	2:44:49
1500	4:30:18	4:35:49	5:06:12	5:07:51	42:15:08	39:21:22	3:05:00	2:49:45
2000	4:47:47	4:51:18	5:33:13	5:26:22	47:52:20	52:18:22	2:47:26	2:34:20
2500	4:52:41	5:11:52	6:12:23	5:58:30	70:18:28	85:48:18	2:47:30	2:44:56
3000	5:26:48	5:22:10	6:37:39	6:27:44	79:03:15	85:18:46	2:44:08	2:43:15

Table 5.17.: Timing results.

In general we found that the tile-based approach requires more processing time when the vocabulary size is increased. Moreover, the SVM classifiers are by far the slowest, with an maximum processing time of approximately 86 hours. If we compare this to the other methods, which consume 6 hours and 37 minutes at most, it is a significant difference. Furthermore, the histogram tracking approach does not seem to be effected by an increase of the vocabulary size, and on average requires the least amount of time to process the complete dataset.

Chapter 6

Conclusion

This chapter is divided into four parts. First, the conclusions related to object classification are presented in section 6.1. Those associated with object localization are discussed in section 6.2. Finally, section 6.3 presents conclusions concerning the visual vocabulary framework. In addition to conclusions, some suggestions for future research, and ways to possibly improve the presented algorithms, are made in section 6.4.

6.1. Classification

We were able to use the visual vocabulary framework to successfully classify object classes. We have shown that the SVM classifiers have the highest performance with a classification accuracy of 89.1%, which is similar to that of state of the art techniques. The generative model introduced in [7] showed a considerably lower accuracy of 83%. The simplified version of the naive Bayes model resulted in the lowest performance, with an optimal accuracy of 82.2%.

Furthermore, the experiments with the different fusion approaches show that late fusion should be preferred above early fusion. In almost every experiment, we recorded a higher classification accuracy using a vocabulary constructed with the late fusion approach. This was to be expected, since the late fusion is able to create specific visual words for each of the various classes. This could be viewed as describing objects in a separate jargon.

Another observation that can be made is the following. If we look at the naive Bayes models, we can observe an increase of classification accuracy when the vocabulary size is increased. However, the SVM models do not show the same increase, but actually show a slight decrease of accuracy. The explanation of this phenomenon is overfitting. The difference between SVM and the naive Bayes models is that SVM is a discriminative model, while naive Bayes is a generative model. In other words, the SVM model will find a hyperplane that stretches across every dimension (every visual word), even the dimensions which contribute to noise. This implies that classification could well be based on the dimension with the most noise, i.e. the weakest link. Thus, introducing more dimensions can result in a decrease in performance. Generative models are less prone to noise, especially the naive Bayes models, since the noise will be aggregated throughout all object models. In other words, a noisy dimension will just mean that the conditional class probability will go down. However, since the probability is aggregated over all object classes, the class conditional probability will go down for each of the object classes, so it will not have great impact on the decision boundary. What is more, adding a discriminating dimension increases the performance of the generative model, even if it is in combination with noisy dimensions. However, this does not mean that we can indefinitely increase the vocabulary size, since eventually the accuracy will go down due to sparseness in the training data.

6.2. Localization

We have extended the visual vocabulary approach of J. Winn, A. Criminisi and T. Minka, in order to localize instances of object classes. Additionally, we have introduced a novel localization technique, were we optimize certain starting regions by applying histogram tracking. Since histogram tracking uses spatial kernels, we have introduced a loose spatial relation in the *bag of keypoints* approach and introduced a framework in which classification, localization and tracking can be combined. Furthermore, we have compared this novel approach to the most common approach for object localization, the tile-based approach. This approach uses the same classifiers as in the classification process, so we have utilized all of the classification techniques discussed in chapter 3 for this purpose.

The localization results show that the tile-based approach yields the highest performance, when it is applied in combination with the SVM classifiers. The SVM classifiers excel in the localization of all object classes. With an average precision of 65% for the cars class, 65% for the airplanes class, 91.5% for the motorbikes class and 94.1% for the faces class, the performance is well above all other localization algorithms. The simplified naive Bayes model showed the worst performance among the four localization methods. The histogram tracking approach and the tile-based approach using the naive Bayes model showed similar results. Furthermore, we have found that the late fusion approach has a higher performance than the early fusion approach, which is in line with the classification experiments.

Another important aspect of the localization approaches is their computational complexity. For each localization algorithm we measured the time it took to process the complete Caltech4 dataset. As was mentioned, the timing results are solely based on the localization process, because we excluded the time it took to extract the descriptor histograms. Although the tile-based approach has shown the best performance, it has proven to be very inefficient. The tile-based approach in combination with the SVM classifiers can take up to 86 hours to run on the whole dataset. Although this is the worst case scenario, using the largest vocabularies, it takes 7 hours at least, using the smallest vocabularies. On top of that, the timing results are biased by the fact that we did not evaluate every possible region and did not evaluate each tile on every pixel location. If we were to increase the number of tiles or pixel locations in the evaluation process, the timing results would increase rapidly. The histogram tracking approach might not have resulted in the highest detection score, but was far less slow than the tile-based approach, taking an average time of approximately 2 hours and 50 minutes to run on the complete dataset. What is more, the vocabulary size does not affect the histogram tracking approach to the same degree that it affects the tile-based approach.

If we compare the results between the tile-based approach using the simplified naive Bayes model and the histogram tracking approach, we can see that the novel approach usually has a higher average precision. Since both methods use the same object model, it implies that the use of dynamic regions, used in the histogram tracking approach, result in a higher performance than the use of static regions, used in the tile-based approach. The increase in performance can also be explained by the fact that the histogram tracking approach smoothes the pdf, so it is less prone to be affected by irregular peaks in the pdf. On the tile-based approach however, peaks in the pdf will play a crucial role. There might be a risk that the tile-based approach will prefer parts of an object over the whole object, when these parts are the most discriminating, i.e. represent peaks in the pdf. Another drawback of the tile-based approach in respect to the histogram tracking approach, is the selection of appropriate tiles. It is not clear how we should determine the optimal shapes, scales and positions of the 'to-be-evaluated' regions. The selection procedure we applied, which makes use of the mean and variance of the boundingboxes, seems to work. However, it is not invariant to geometric transformations such as rotation.

Furthermore, the localization experiments show that for the generative object models, large vocabularies do not always results in a higher average precision. Nevertheless, we saw that larger vocabularies do indeed improve precision, but only in the segment of detections with the highest confidence. This suggests that a higher expressiveness is, to some degree, beneficial for the accuracy of the detection, while being less good in detecting every single object. This conclusion is also supported by the classification results in section 5.4, where we saw that the accuracy of classification also increased with a larger dictionary size.

If we compare the classification results with the localization results, we see some contradictions. We found that the classification accuracy of the naive Bayes models goes up when we increase the vocabulary size. However, the localization experiments shows the opposite. In general, localization using smaller vocabularies result in a higher average precision. So, even though the object models learned with use of larger vocabularies are more capable to distinguish one object class from other object classes, they are less suited for localization of objects. Again, we are confronted with overfitting. It appears that some models with high classification accuracy, do not generalize the appearance of objects, because the detection performance is lower. We believe this is because the localization algorithm will focus on the most discriminating parts of an object, thus a risk of miss detection exists. For example, if we assume that the most discriminating part of a motorbike is the wheel, then there is a risk that the only region that will be detected is the region containing the wheel, in stead of detecting the whole object. The general consensus is that an increase of classification accuracy means that the object models have a better generalization. However, we believe this is not true.

6.3. Visual Vocabulary Framework

We have used the visual vocabulary framework successfully for both object recognition and localization. It is a very simple but effective framework for the extraction of image statistics. Furthermore, this framework allows us to detect, classify and track object classes using the same techniques. However, we have found that the visual vocabulary framework is prone to overfitting and it is hard to say which visual vocabulary is best suited for a given task. From the experiments we can not clearly infer a general rule of thumb for determining the optimal vocabulary size or fusion approach. We can say that in most cases, the late fusion approach yields higher results. We can also comment on the vocabulary size. Smaller vocabulary sizes will always decrease the complexity of the descriptor extraction process, so smaller vocabularies should be preferred when it comes to efficiency.

6.4. Future Work

We have found that the visual vocabulary framework can be applied in many different research area's. Further research in any of these area's is possible, and it would be interesting to investigate a possible integration of different applications. One of those integrations would be that of tracking and classification. In the framework we have proposed, it is possible to integrate the two tasks and there are many applications feasible that would benefit from both tracking and object recognition.

Another interesting direction of research is the investigation of vocabulary pruning. There might be a way to not only determine the appropriate vocabulary size, but also determine the appropriate visual words. One could think of feature selection methods to accomplish this goal. The use of vocabulary pruning would increase both performance and efficiency. A performance increase would be visible due to a decrease the amount of noise introduced in the framework. The benefit of vocabulary pruning to efficiency is trivial, i.e. smaller vocabulary means less computational effort.

We saw that the use of dynamic regions is beneficial to performance, as well as the efficiency. However, the novel localization algorithm does not yield results that can compete with the tile-based approach. Maybe we could improve the histogram tracking approach by using different similarity measures. The naive Bayes model introduced by Winn [7] might be applicable in this case.

Additionally, we could think of using contextual information to guide the localization task. In the approach we chosen, contextual information is ignored, with the risk of low accuracy when local image information is insufficient (occlusion and truncation, small objects). It would be nice to see how contextual information would influence the localization results.

Another ambitious future research project is the categorization of scenes. We can use the information collected by our object detectors to build scene models, which enable us to extract even higher level semantical information of images. It would bring us a step further to reaching our goal, which is to bridge the semantic gap.

Appendix A

SVM Results

During the evaluation of the classification accuracy for SVM classifiers, we have tried several parameter settings. Here, the results for all the different parameter settings are presented. The parameters we have applied in our experiments were

$$\begin{aligned} C &= \{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100\} \\ \gamma &= \{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100\} \end{aligned}$$

Tables A.1 through A.8 show the classification accuracy for the different parameter settings using the early fusion, in a numerical manner. Similarly, tables A.9 through A.16 contain the results for late fusion.

	γ										
costs	0.001	0.005	0.01	0.05	0.1	0.5	1	5	10	50	100
0.001	0.373	0.439	0.553	0.800	0.813	0.828	0.825	0.825	0.825	0.825	0.830
0.005	0.435	0.800	0.789	0.824	0.819	0.825	0.825	0.827	0.825	0.832	0.842
0.01	0.584	0.810	0.806	0.824	0.824	0.824	0.825	0.825	0.830	0.842	0.848
0.05	0.806	0.826	0.825	0.826	0.824	0.827	0.825	0.834	0.842	0.865	0.870
0.1	0.814	0.823	0.826	0.825	0.825	0.825	0.832	0.843	0.849	0.870	0.878
0.5	0.823	0.828	0.828	0.829	0.828	0.839	0.843	0.866	0.873	0.888	0.890
1	0.828	0.831	0.832	0.831	0.834	0.842	0.853	0.874	0.883	0.891*	0.890
5	0.829	0.828	0.829	0.843	0.850	0.867	0.879	0.881	0.885	0.889	0.887
10	0.828	0.828	0.829	0.850	0.855	0.874	0.881	0.885	0.886	0.884	0.885
50	0.830	0.831	0.830	0.833	0.842	0.860	0.863	0.866	0.864	0.864	0.865
100	0.801	0.817	0.817	0.815	0.822	0.835	0.844	0.845	0.844	0.845	0.845

Table A.1.: Classification accuracy for the different parameter settings, using a vocabulary size of 100 and early fusion.

	γ										
costs	0.001	0.005	0.01	0.05	0.1	0.5	1	5	10	50	100
0.001	0.373	0.376	0.461	0.806	0.826	0.829	0.831	0.832	0.832	0.834	0.836
0.005	0.374	0.729	0.806	0.813	0.823	0.832	0.833	0.833	0.833	0.842	0.844
0.01	0.447	0.803	0.822	0.825	0.829	0.831	0.832	0.833	0.835	0.844	0.849
0.05	0.813	0.822	0.825	0.832	0.833	0.833	0.834	0.842	0.843	0.861	0.868
0.1	0.815	0.828	0.831	0.833	0.834	0.834	0.836	0.844	0.851	0.868	0.876
0.5	0.828	0.837	0.837	0.837	0.836	0.846	0.849	0.861	0.866	0.883	0.886
1	0.836	0.839	0.840	0.841	0.840	0.853	0.855	0.865	0.875	0.887*	0.886
5	0.843	0.843	0.843	0.855	0.857	0.861	0.866	0.878	0.877	0.881	0.881
10	0.844	0.844	0.844	0.859	0.862	0.865	0.875	0.876	0.875	0.875	0.875
50	0.816	0.815	0.814	0.820	0.828	0.858	0.862	0.862	0.861	0.862	0.862
100	0.799	0.807	0.806	0.809	0.814	0.831	0.844	0.847	0.847	0.846	0.846

Table A.2.: Classification accuracy for the different parameter settings, using a vocabulary size of 250 and early fusion.

	γ										
costs	0.001	0.005	0.01	0.05	0.1	0.5	1	5	10	50	100
0.001	0.373	0.373	0.392	0.716	0.815	0.819	0.825	0.839	0.840	0.840	0.844
0.005	0.373	0.595	0.716	0.814	0.825	0.840	0.838	0.839	0.841	0.842	0.842
0.01	0.393	0.723	0.800	0.812	0.822	0.840	0.840	0.841	0.844	0.842	0.845
0.05	0.706	0.815	0.817	0.838	0.839	0.840	0.842	0.842	0.842	0.855	0.860
0.1	0.814	0.818	0.836	0.841	0.841	0.842	0.844	0.842	0.847	0.862	0.871
0.5	0.820	0.839	0.842	0.845	0.847	0.846	0.847	0.858	0.863	0.882	0.890
1	0.834	0.846	0.847	0.849	0.848	0.847	0.850	0.862	0.869	0.890	0.885
5	0.848	0.851	0.850	0.852	0.852	0.860	0.863	0.889*	0.885	0.886	0.885
10	0.850	0.852	0.852	0.856	0.854	0.865	0.873	0.887	0.885	0.885	0.886
50	0.813	0.814	0.813	0.818	0.829	0.868	0.877	0.875	0.875	0.876	0.875
100	0.786	0.786	0.786	0.787	0.796	0.839	0.852	0.856	0.857	0.857	0.855

Table A.3.: Classification accuracy for the different parameter settings, using a vocabulary size of 500 and early fusion.

	γ										
costs	0.001	0.005	0.01	0.05	0.1	0.5	1	5	10	50	100
0.001	0.373	0.373	0.373	0.641	0.720	0.818	0.812	0.851	0.853	0.855	0.855
0.005	0.373	0.474	0.666	0.818	0.819	0.850	0.851	0.854	0.854	0.860	0.857
0.01	0.373	0.637	0.736	0.827	0.830	0.851	0.855	0.854	0.856	0.857	0.855
0.05	0.638	0.822	0.828	0.851	0.852	0.854	0.855	0.859	0.857	0.858	0.862
0.1	0.731	0.821	0.822	0.853	0.854	0.855	0.856	0.857	0.855	0.863	0.866
0.5	0.808	0.852	0.857	0.857	0.857	0.862	0.860	0.860	0.864	0.876	0.879
1	0.840	0.858	0.859	0.860	0.860	0.860	0.857	0.863	0.869	0.881	0.880
5	0.856	0.861	0.860	0.862	0.861	0.861	0.862	0.876	0.881	0.882*	0.881
10	0.856	0.856	0.858	0.859	0.860	0.863	0.867	0.881	0.884	0.881	0.881
50	0.812	0.812	0.812	0.815	0.825	0.854	0.870	0.868	0.870	0.870	0.869
100	0.776	0.776	0.776	0.771	0.784	0.829	0.852	0.855	0.855	0.855	0.855

Table A.4.: Classification accuracy for the different parameter settings, using a vocabulary size of 1000 and early fusion.

	γ										
costs	0.001	0.005	0.01	0.05	0.1	0.5	1	5	10	50	100
0.001	0.373	0.373	0.373	0.510	0.660	0.817	0.830	0.852	0.852	0.854	0.855
0.005	0.373	0.423	0.522	0.790	0.809	0.849	0.852	0.852	0.854	0.855	0.854
0.01	0.373	0.519	0.671	0.811	0.806	0.848	0.851	0.854	0.855	0.855	0.856
0.05	0.518	0.804	0.824	0.844	0.848	0.852	0.854	0.856	0.854	0.860	0.861
0.1	0.686	0.834	0.819	0.852	0.853	0.854	0.855	0.855	0.856	0.861	0.866
0.5	0.801	0.846	0.854	0.857	0.858	0.858	0.857	0.859	0.862	0.869	0.876
1	0.834	0.858	0.857	0.860	0.859	0.858	0.857	0.863	0.867	0.877	0.878
5	0.856	0.861	0.862	0.861	0.861	0.863	0.864	0.870	0.875	0.886	0.885
10	0.854	0.859	0.857	0.861	0.861	0.862	0.863	0.877	0.885	0.888*	0.887
50	0.811	0.811	0.810	0.815	0.824	0.848	0.860	0.870	0.871	0.871	0.870
100	0.778	0.775	0.775	0.773	0.783	0.826	0.848	0.852	0.853	0.853	0.853

Table A.5.: Classification accuracy for the different parameter settings, using a vocabulary size of 1500 and early fusion.

	γ										
costs	0.001	0.005	0.01	0.05	0.1	0.5	1	5	10	50	100
0.001	0.373	0.373	0.373	0.509	0.656	0.809	0.798	0.851	0.852	0.854	0.852
0.005	0.373	0.384	0.482	0.779	0.798	0.839	0.850	0.852	0.853	0.856	0.856
0.01	0.373	0.490	0.671	0.818	0.830	0.848	0.853	0.853	0.853	0.856	0.856
0.05	0.514	0.768	0.804	0.842	0.851	0.853	0.853	0.856	0.857	0.855	0.859
0.1	0.669	0.798	0.812	0.852	0.853	0.854	0.854	0.856	0.856	0.858	0.861
0.5	0.779	0.835	0.852	0.855	0.856	0.857	0.856	0.855	0.860	0.866	0.870
1	0.799	0.851	0.855	0.857	0.856	0.859	0.856	0.859	0.864	0.870	0.879
5	0.855	0.860	0.860	0.857	0.860	0.857	0.858	0.867	0.872	0.880	0.885*
10	0.858	0.861	0.861	0.860	0.857	0.858	0.862	0.873	0.880	0.885*	0.884
50	0.813	0.814	0.813	0.817	0.825	0.848	0.854	0.869	0.866	0.866	0.868
100	0.781	0.780	0.780	0.781	0.788	0.828	0.843	0.850	0.849	0.850	0.851

Table A.6.: Classification accuracy for the different parameter settings, using a vocabulary size of 2000 and early fusion.

	γ										
costs	0.001	0.005	0.01	0.05	0.1	0.5	1	5	10	50	100
0.001	0.373	0.373	0.373	0.501	0.578	0.796	0.776	0.849	0.850	0.850	0.851
0.005	0.373	0.388	0.467	0.732	0.807	0.831	0.852	0.850	0.850	0.852	0.853
0.01	0.373	0.484	0.600	0.810	0.815	0.845	0.850	0.850	0.852	0.853	0.851
0.05	0.498	0.725	0.807	0.822	0.850	0.850	0.849	0.852	0.853	0.853	0.855
0.1	0.596	0.797	0.805	0.845	0.851	0.850	0.851	0.854	0.853	0.855	0.862
0.5	0.764	0.823	0.850	0.854	0.854	0.854	0.857	0.854	0.858	0.863	0.867
1	0.784	0.849	0.854	0.857	0.857	0.858	0.855	0.858	0.860	0.869	0.878
5	0.846	0.860	0.859	0.858	0.857	0.857	0.859	0.866	0.870	0.881*	0.878
10	0.855	0.858	0.860	0.858	0.857	0.861	0.860	0.866	0.875	0.879	0.876
50	0.815	0.816	0.816	0.819	0.825	0.848	0.861	0.876	0.876	0.876	0.876
100	0.783	0.782	0.781	0.783	0.789	0.822	0.837	0.851	0.850	0.851	0.850

Table A.7.: Classification accuracy for the different parameter settings, using a vocabulary size of 2500 and early fusion.

	γ										
costs	0.001	0.005	0.01	0.05	0.1	0.5	1	5	10	50	100
0.001	0.373	0.373	0.373	0.465	0.535	0.784	0.751	0.852	0.857	0.857	0.858
0.005	0.373	0.388	0.446	0.704	0.773	0.831	0.852	0.856	0.858	0.859	0.858
0.01	0.373	0.471	0.550	0.787	0.804	0.850	0.857	0.858	0.858	0.857	0.856
0.05	0.473	0.688	0.774	0.842	0.851	0.857	0.857	0.859	0.857	0.854	0.860
0.1	0.556	0.790	0.801	0.855	0.857	0.857	0.859	0.858	0.857	0.860	0.860
0.5	0.759	0.825	0.854	0.858	0.858	0.858	0.859	0.855	0.858	0.862	0.861
1	0.778	0.844	0.855	0.857	0.860	0.860	0.857	0.859	0.857	0.859	0.869
5	0.841	0.859	0.858	0.860	0.859	0.855	0.860	0.862	0.865	0.873	0.876
10	0.856	0.857	0.859	0.858	0.856	0.857	0.860	0.861	0.869	0.877*	0.873
50	0.822	0.822	0.822	0.825	0.828	0.851	0.855	0.871	0.869	0.869	0.869
100	0.787	0.785	0.785	0.789	0.796	0.825	0.840	0.851	0.850	0.852	0.851

Table A.8.: Classification accuracy for the different parameter settings, using a vocabulary size of 3000 and early fusion.

	γ										
costs	0.001	0.005	0.01	0.05	0.1	0.5	1	5	10	50	100
0.001	0.373	0.448	0.511	0.811	0.829	0.830	0.831	0.827	0.828	0.831	0.841
0.005	0.438	0.806	0.807	0.816	0.829	0.827	0.827	0.828	0.832	0.841	0.851
0.01	0.519	0.807	0.824	0.827	0.830	0.828	0.828	0.832	0.840	0.851	0.859
0.05	0.816	0.820	0.828	0.829	0.829	0.829	0.833	0.841	0.852	0.866	0.872
0.1	0.821	0.829	0.830	0.828	0.829	0.833	0.841	0.852	0.858	0.874	0.876
0.5	0.831	0.833	0.833	0.833	0.836	0.843	0.855	0.869	0.881	0.884	0.888
1	0.836	0.836	0.836	0.839	0.845	0.854	0.861	0.881	0.884	0.890*	0.889
5	0.837	0.837	0.838	0.852	0.858	0.873	0.882	0.888	0.888	0.886	0.886
10	0.842	0.842	0.842	0.856	0.863	0.885	0.882	0.888	0.887	0.884	0.884
50	0.837	0.837	0.837	0.837	0.842	0.866	0.877	0.879	0.879	0.879	0.879
100	0.809	0.821	0.822	0.822	0.825	0.844	0.862	0.862	0.859	0.861	0.862

Table A.9.: Classification accuracy for the different parameter settings, using a vocabulary size of 100 and late fusion.

	γ										
costs	0.001	0.005	0.01	0.05	0.1	0.5	1	5	10	50	100
0.001	0.373	0.373	0.444	0.776	0.813	0.836	0.833	0.840	0.840	0.840	0.840
0.005	0.373	0.703	0.784	0.822	0.817	0.838	0.840	0.840	0.840	0.846	0.848
0.01	0.459	0.767	0.814	0.826	0.838	0.840	0.840	0.839	0.840	0.848	0.850
0.05	0.777	0.829	0.830	0.840	0.841	0.841	0.840	0.846	0.849	0.854	0.863
0.1	0.817	0.821	0.834	0.841	0.841	0.840	0.840	0.850	0.851	0.865	0.877
0.5	0.826	0.841	0.843	0.842	0.842	0.845	0.850	0.858	0.867	0.880	0.884
1	0.838	0.845	0.845	0.845	0.844	0.853	0.855	0.869	0.879	0.886	0.891*
5	0.837	0.839	0.840	0.851	0.858	0.861	0.865	0.878	0.883	0.884	0.886
10	0.838	0.838	0.838	0.859	0.859	0.863	0.874	0.880	0.882	0.882	0.883
50	0.830	0.832	0.832	0.826	0.829	0.865	0.866	0.870	0.868	0.869	0.870
100	0.799	0.804	0.805	0.804	0.805	0.839	0.850	0.854	0.852	0.852	0.853

Table A.10.: Classification accuracy for the different parameter settings, using a vocabulary size of 250 and late fusion.

	γ										
costs	0.001	0.005	0.01	0.05	0.1	0.5	1	5	10	50	100
0.001	0.373	0.373	0.410	0.722	0.795	0.828	0.830	0.842	0.841	0.843	0.844
0.005	0.373	0.533	0.710	0.828	0.812	0.839	0.842	0.841	0.842	0.848	0.852
0.01	0.400	0.712	0.800	0.823	0.840	0.840	0.840	0.841	0.843	0.852	0.852
0.05	0.728	0.824	0.828	0.836	0.841	0.841	0.842	0.848	0.852	0.858	0.863
0.1	0.798	0.821	0.832	0.839	0.841	0.842	0.844	0.852	0.854	0.864	0.870
0.5	0.817	0.839	0.841	0.843	0.842	0.847	0.853	0.857	0.866	0.880	0.878
1	0.833	0.846	0.846	0.846	0.845	0.855	0.858	0.865	0.871	0.879	0.879
5	0.848	0.852	0.850	0.853	0.861	0.862	0.866	0.879	0.879	0.881	0.881
10	0.850	0.852	0.850	0.862	0.865	0.868	0.871	0.880	0.882*	0.882*	0.884
50	0.825	0.826	0.825	0.832	0.837	0.868	0.873	0.875	0.873	0.873	0.874
100	0.805	0.804	0.803	0.804	0.809	0.847	0.860	0.858	0.857	0.858	0.858

Table A.11.: Classification accuracy for the different parameter settings, using a vocabulary size of 500 and late fusion.

	γ										
costs	0.001	0.005	0.01	0.05	0.1	0.5	1	5	10	50	100
0.001	0.373	0.373	0.376	0.603	0.755	0.826	0.824	0.844	0.844	0.847	0.847
0.005	0.373	0.464	0.560	0.818	0.816	0.838	0.844	0.845	0.846	0.853	0.853
0.01	0.374	0.579	0.737	0.825	0.834	0.840	0.844	0.845	0.847	0.854	0.855
0.05	0.606	0.804	0.817	0.839	0.844	0.845	0.845	0.852	0.854	0.857	0.862
0.1	0.745	0.809	0.825	0.842	0.845	0.846	0.847	0.854	0.857	0.862	0.869
0.5	0.803	0.838	0.846	0.847	0.849	0.851	0.855	0.860	0.866	0.883	0.885
1	0.823	0.846	0.849	0.850	0.851	0.857	0.858	0.867	0.871	0.886	0.885
5	0.848	0.850	0.850	0.856	0.860	0.866	0.865	0.882	0.887	0.886	0.886
10	0.853	0.854	0.854	0.858	0.862	0.865	0.869	0.889*	0.884	0.885	0.884
50	0.829	0.828	0.829	0.832	0.836	0.862	0.880	0.882	0.882	0.882	0.882
100	0.795	0.796	0.796	0.792	0.800	0.842	0.862	0.864	0.864	0.865	0.864

Table A.12.: Classification accuracy for the different parameter settings, using a vocabulary size of 1000 and late fusion.

	γ										
costs	0.001	0.005	0.01	0.05	0.1	0.5	1	5	10	50	100
0.001	0.373	0.373	0.373	0.513	0.670	0.819	0.815	0.848	0.845	0.847	0.847
0.005	0.373	0.437	0.487	0.812	0.801	0.841	0.846	0.846	0.847	0.852	0.852
0.01	0.373	0.497	0.682	0.818	0.840	0.846	0.848	0.848	0.848	0.852	0.849
0.05	0.515	0.793	0.810	0.836	0.845	0.847	0.846	0.852	0.852	0.856	0.859
0.1	0.691	0.798	0.824	0.849	0.847	0.848	0.849	0.853	0.849	0.859	0.868
0.5	0.788	0.831	0.851	0.852	0.852	0.852	0.853	0.855	0.859	0.873	0.883
1	0.811	0.848	0.850	0.854	0.853	0.852	0.855	0.859	0.868	0.885	0.885
5	0.844	0.853	0.854	0.858	0.860	0.860	0.860	0.873	0.882	0.887*	0.887*
10	0.853	0.854	0.854	0.857	0.862	0.863	0.865	0.880	0.885	0.885	0.884
50	0.829	0.830	0.830	0.834	0.839	0.858	0.871	0.878	0.876	0.876	0.877
100	0.797	0.799	0.798	0.795	0.807	0.838	0.860	0.867	0.866	0.866	0.866

Table A.13.: Classification accuracy for the different parameter settings, using a vocabulary size of 1500 and late fusion.

	γ										
costs	0.001	0.005	0.01	0.05	0.1	0.5	1	5	10	50	100
0.001	0.373	0.373	0.373	0.484	0.635	0.821	0.817	0.849	0.847	0.848	0.848
0.005	0.373	0.395	0.483	0.751	0.831	0.831	0.849	0.847	0.847	0.849	0.851
0.01	0.373	0.470	0.623	0.830	0.814	0.848	0.846	0.845	0.848	0.852	0.850
0.05	0.488	0.766	0.812	0.833	0.848	0.849	0.849	0.849	0.852	0.853	0.857
0.1	0.634	0.815	0.814	0.844	0.846	0.847	0.850	0.854	0.852	0.858	0.861
0.5	0.793	0.843	0.846	0.852	0.854	0.852	0.853	0.855	0.859	0.870	0.880
1	0.799	0.849	0.851	0.854	0.855	0.854	0.850	0.858	0.863	0.878	0.880
5	0.843	0.855	0.854	0.857	0.860	0.857	0.861	0.870	0.880	0.883	0.883
10	0.850	0.854	0.854	0.859	0.861	0.860	0.862	0.879	0.884	0.884*	0.881
50	0.829	0.832	0.832	0.836	0.839	0.855	0.868	0.876	0.877	0.876	0.875
100	0.804	0.804	0.804	0.800	0.805	0.836	0.853	0.859	0.858	0.858	0.861

Table A.14.: Classification accuracy for the different parameter settings, using a vocabulary size of 2000 and late fusion.

	γ										
costs	0.001	0.005	0.01	0.05	0.1	0.5	1	5	10	50	100
0.001	0.373	0.373	0.373	0.460	0.559	0.804	0.807	0.848	0.844	0.846	0.847
0.005	0.373	0.389	0.467	0.731	0.818	0.838	0.847	0.847	0.846	0.847	0.850
0.01	0.373	0.447	0.551	0.815	0.815	0.848	0.845	0.846	0.846	0.850	0.850
0.05	0.475	0.757	0.808	0.839	0.847	0.846	0.846	0.847	0.850	0.856	0.859
0.1	0.589	0.810	0.809	0.845	0.845	0.847	0.847	0.849	0.852	0.858	0.862
0.5	0.780	0.825	0.839	0.848	0.850	0.849	0.850	0.856	0.860	0.871	0.876
1	0.799	0.831	0.855	0.850	0.857	0.853	0.853	0.860	0.865	0.876	0.882
5	0.846	0.857	0.858	0.859	0.860	0.858	0.861	0.872	0.877	0.884	0.881
10	0.852	0.855	0.856	0.858	0.862	0.866	0.867	0.881	0.882	0.886*	0.885
50	0.833	0.834	0.835	0.838	0.835	0.853	0.863	0.874	0.874	0.873	0.873
100	0.800	0.805	0.804	0.803	0.812	0.836	0.854	0.863	0.863	0.864	0.864

Table A.15.: Classification accuracy for the different parameter settings, using a vocabulary size of 2500 and late fusion.

	γ										
costs	0.001	0.005	0.01	0.05	0.1	0.5	1	5	10	50	100
0.001	0.373	0.373	0.373	0.463	0.526	0.783	0.748	0.846	0.847	0.848	0.847
0.005	0.373	0.385	0.447	0.760	0.781	0.828	0.848	0.847	0.849	0.853	0.856
0.01	0.373	0.475	0.524	0.799	0.785	0.846	0.846	0.849	0.850	0.855	0.854
0.05	0.471	0.737	0.777	0.829	0.842	0.849	0.850	0.853	0.856	0.854	0.861
0.1	0.543	0.786	0.806	0.847	0.849	0.848	0.851	0.856	0.854	0.862	0.864
0.5	0.742	0.826	0.846	0.852	0.854	0.854	0.855	0.856	0.861	0.871	0.874
1	0.764	0.846	0.854	0.857	0.857	0.856	0.858	0.861	0.866	0.873	0.876
5	0.837	0.858	0.858	0.858	0.861	0.860	0.861	0.873	0.875	0.880	0.880
10	0.849	0.856	0.855	0.858	0.860	0.861	0.866	0.878	0.880	0.880	0.882*
50	0.829	0.831	0.831	0.840	0.840	0.860	0.866	0.882*	0.880	0.879	0.880
100	0.799	0.798	0.798	0.803	0.805	0.834	0.850	0.861	0.861	0.861	0.860

Table A.16.: Classification accuracy for the different parameter settings, using a vocabulary size of 3000 and late fusion.

Bibliography

[1] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(12):1349–1380, 2000.

[2] Michael C. Burl, Markus Weber, and Pietro Perona. A probabilistic approach to object recognition using local photometry and global geometry. *Lecture Notes in Computer Science*, 1407:628–641, 1998.

[3] M. Weber, M. Welling, and P. Perona. Towards automatic discovery of object categories. pages II: 101–108, 2000.

[4] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003.

[5] C. Dance, J. Willamowski, L. Fan, C. Bray, and G. Csurka. Visual categorization with bags of keypoints. In *ECCV International Workshop on Statistical Learning in Computer Vision*, 2004.

[6] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *ECCV (1)*, pages 128–142, 2002.

[7] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision*, pages 1800–1807. IEEE Computer Society, 2005.

[8] T.K. Leung and J. Malik. Recognizing surfaces using three-dimensional textons. In *ICCV (2)*, pages 1010–1017, 1999.

[9] T.K. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *Int. J. Comput. Vision*, 43(1):29–44, 2001.

[10] D. Hoiem, A.A. Efros, and M. Hebert. Putting objects in perspective. In *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 2137 – 2144, June 2006.

[11] A. Agarwal and W. Triggs. Hyperfeatures: Multilevel local coding for visual recognition. *ECCV International Workshop on Statistical Learning in Computer Vision*, 2006.

[12] K. Murphy, A. Torralba, D. Eaton, and W. Freeman. Object detection and localization using local and global features. In *Toward Category-Level Object Recognition*, pages 382–400, 2006.

[13] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, 2001.

[14] I. Laptev. Improvements of object detection using boosted histograms. *Proc. British Machine Vision Conf. (BMVC)*, 2006.

[15] J. Sivic, B. Russell, A.A. Efros, A. Zisserman, and B. Freeman. Discovering objects and their location in images. In *International Conference on Computer Vision (ICCV 2005)*, October 2005.

[16] C. Schmid. Constructing models for content-based image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Kauai, Hawaii, USA*, 2001.

[17] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. *Int. J. Comput. Vision*, 62(1-2):61–81, 2005.

[18] A.R. Webb. *Statistical Pattern Recognition, 2nd Edition*. John Wiley & Sons, October 2002.

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

[20] J. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. pages 61–74, 2000.

[21] P. Fryzlewicz and G.P. Nason. A haar-fisz algorithm for poisson intensity estimation. *J. Comp. Graph. Stat.*

[22] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(5):564–577, 2003.

[23] E. Han and G. Karypis. Centroid-based document classification: Analysis and experimental results. In *Principles of Data Mining and Knowledge Discovery*, pages 424–431, 2000.

[24] V. Tam, A. Santoso, and R. Setiono. A comparative study of centroid-based, neighborhood-based and statistical approaches for effective document categorization. *ICPR*, 4:235–238, 2002.

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

[26] D. Comaniciu and P. Meer. Mean shift analysis and applications. In *ICCV (2)*, pages 1197–1203, 1999.

[27] Z. Zivkovic and B. Krose. An EM-like algorithm for color-histogram-based object tracking. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2004.

[28] T. Minka. The ‘summation hack’ as an outlier model.

[29] J.L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[30] J.L. Bentley. Multidimensional binary search trees in database applications. *IEEE Trans. Software Eng.*, 5(4):333–340, 1979.

[31] D. Pelleg and A. Moore. Accelerating exact k-means algorithms with geometric reasoning. In *Proceedings of the Fifth International Conference on Knowledge Discovery in Databases*, pages 277–281. AAAI Press, 1999.