Visual Word Ambiguity

J.C. van Gemert, C.J. Veenman, A.W.M. Smeulders, J.M. Geusebroek

***Abstract*— This paper studies automatic image classification by modeling soft-assignment in the popular codebook model. The codebook model describes an image as a bag of discrete visual words selected from a vocabulary, where the frequency distributions of visual words in an image allow classification. One inherent component of the codebook model is the assignment of discrete visual words to continuous image features. Despite the clear mismatch of this hard assignment with the nature of continuous features, the approach has been applied successfully for some years. In this paper we investigate four types of soft-assignment of visual words to image features. We demonstrate that explicitly modeling visual word assignment ambiguity improves classification performance compared to the hard-assignment of the traditional codebook model. The traditional codebook model is compared against our method for five well-known datasets: 15 natural scenes, Caltech-101, Caltech-256, and Pascal VOC 2007/2008. We demonstrate that large codebook vocabulary sizes completely deteriorate the performance of the traditional model, whereas the proposed model performs consistently. Moreover, we show that our method profits in high-dimensional feature spaces and reaps higher benefits when increasing the number of image categories.**

***Index Terms*— Computer vision, Object recognition, Image/video retrieval.**

I. Introduction

VERBAL descriptions of visual characteristics like a color or a texture are often ambiguous. For example, quantifying a texture with *"A predominantly smooth yellowish-red surface with a few cracks"* leaves considerable room for interpretation. One of the interpretations of the popular codebook model for automatic image classification [1]–[31] is that it expresses images in terms of visual words. The model represents high-dimensional image features by discrete and disjunct visual prototypes that are predefined in a vocabulary. The visual word analogy of the codebook model includes semantic modeling at the word level [6], [10], [18], [29] or at the topic level [1], [4], [6], [8], [13], [23], [26], [30]. Spatial image layout [1], [5], [6], [14], [19], [26] can be seen as modeling phrases, whereas visual vocabulary tuning [12], [13], [15], [21], [28], [30], [31] resembles modeling domain-specific terminology. These models incorporate image-specific properties within the conceptual visual word analogy. In this paper we introduce another aspect of the visual word analogy, namely the use of ambiguous linguistic quantifiers as *"some"*, *"a few"*, *"-ish"*, *"predominantly"*, *"much"*. Without such quantifiers to express ambiguity, the description of the aforementioned texture is reduced to *"A smooth red surface"*. We incorporate ambiguity in the codebook model by smoothly assigning continuous image features to discrete visual words. We show that ambiguity modeling leads to more expressive models that improve classification performance.

One inherent component of the codebook model is the assignment of image feature vectors to visual words in the vocabulary.

Manuscript received June 26 2008.
The authors are with with the University of Amsterdam.

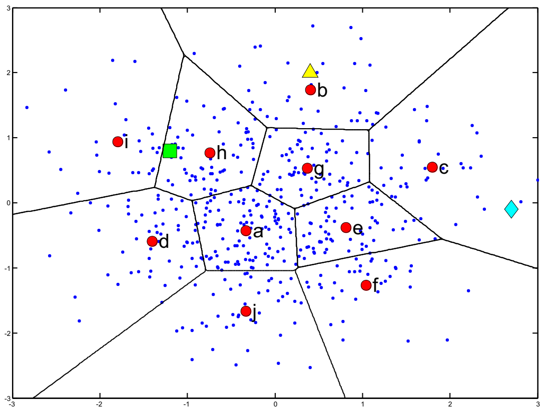


Fig. 1. An example illustrating visual word ambiguity in the codebook model. The small dots represent image feature vectors. The labeled red circles are visual words found by unsupervised clustering. The triangle represents a data sample that is well suited to the codebook model. Visual word uncertainty is exemplified by the square, whereas visual word plausibility is illustrated by the diamond.

Here, an important assumption is that a discrete visual word is a characteristic representative of an image feature. The continuous nature of visual appearance complicates selecting a representative visual word for an image feature. An image feature may have zero, one, or multiple candidates in the visual word vocabulary. With one candidate there is no ambiguity. Selecting a codeword from multiple realistic candidates gives rise to *visual word uncertainty*, whereas *visual word plausibility* refers to selecting a codeword without a suitable candidate in the vocabulary. Figure 1 illustrates these cases. Current methods assume that an image feature is well represented by its single, best representing visual word.

The contribution of this paper is an investigation of visual word ambiguity leading to explicit ambiguity modeling in the codebook model. We investigate the effect of ambiguity modeling on four aspects of the codebook model. First, we investigate the classification performance of various types of ambiguity modeling. Second, we look at vocabulary expressiveness by relating ambiguity modeling and the vocabulary size. Third, we consider the effect of ambiguity on the image feature dimensionality. Our fourth contribution investigates the connection between ambiguity modeling and the number of image categories. All contribution are thoroughly experimentally verified on five well-known image categorization datasets: 15 natural scenes, Caltech-101, Caltech-256, and Pascal VOC 2007/2008. Given the current drive of the state of the art to increase feature dimensionality, vocabulary size, and the number of image categories, we argue that our contributions play an important role in practical image classification.

This paper is organized as follows. The next section discusses the related literature on codebook-based scene classification. Section III introduces ambiguity in the codebook model. We show the performance and consequences of our method on five datasets in section IV, whereas section V concludes the paper. In this paper we use the terms *codeword* and *visual word* interchangeably.

II. Related Work

The visual word vocabulary in the codebook model may be constructed in various ways. Typically, a vocabulary is constructed by applying k-means clustering on image features [4], [14]–[16], [20], [23]–[26], [30], [31]. K-means minimizes the variance between the clusters and the data, placing clusters near the most frequently occurring features. The most frequent features, as noted by Jurie and Triggs [12] and others [3], [23], are not necessarily the most discriminative. The discriminative power of the vocabulary may be improved by alternative clustering algorithms [12], [15], incorporating image class labels [19], [30], [31], or creating specifically tuned vocabularies for each image category as suggested by Perronnin *et al.* [21] and others [13], [28]. In contrast to clustering, a vocabulary may be obtained by manually labeling image patches with a semantic label [6], [10], [18], [29]. For example, Vogel *et al.* [29] construct a vocabulary by labeling image patches of *sky*, *water* or *grass*. The idea behind a semantic vocabulary is that the meaning of an image may be expressed in the meaning of its constituent visual words. Both the semantic and the clustered vocabulary creation methods may reduce visual word ambiguity by carefully selection the vocabulary. For example, when distinguishing a *sunset* from a *forest*, the ambiguity between the colors *pink* and *orange* is irrelevant, since both colors will be absent in a *forest*. Careful vocabulary selection, however, does not address visual word ambiguity itself.

In literature, visual word ambiguity modeling is used occasionally, often ad-hoc motivated, and rarely evaluated. Tuytelaars and Schmid [28] and Jiang *et al.* [11] assign an image feature to visual words that are neighbors in feature space. Alternatively, a probabilistic visual word voting scheme may be used [1], [2], [7], [17], [21], [22]. Here, each image feature contributes to multiple visual words relative to the posterior probability of the image feature given the visual word. Since multiple visual words are being considered, these methods cope with *visual word uncertainty*. These works recognize the importance of visual word uncertainty and show that it leads to increased classification performance. These works, however, lack a clear motivation for their type of ambiguity modeling, and ignore *visual word plausibility*. The plausibility of a visual word is employed by Boiman *et al.* [3] who use the distance to the single best neighbor in feature space. Their method cannot select multiple relevant visual words and therefore does not take *visual word uncertainty* into account. The uncertainty of a visual word as well as its plausibility are used by Jégou *et al.* [32]. The authors weight closer neighbors heavier than farther ones, without normalizing the scores to a posterior probability. Hence, multiple candidates can be selected, and implausible ones are given a low weight. None of these methods provide much motivation or evaluation for their choice of dealing with visual word ambiguity. In this paper, however, we motivate and evaluate several types of visual word ambiguity, extending our preliminary work [9] with state-of-the-art results on two additional datasets, an extensive evaluation of the vocabulary size, and ample extra analysis for all evaluated datasets.

Besides direct ambiguity modeling, ambiguity might be addressed by modeling visual word co-occurrences. Co-occurrence modeling may address ambiguity because it is likely that similar visual words with high ambiguity co-occur frequently. When these ambiguous visual words are grouped together their intra-ambiguity is resolved. Co-occurring visual word modeling is performed after assigning visual words to image features. Typically, co-occurrence is captured with a generative probabilistic model [33]–[35]. A generative codebook model [1], [4], [6], [8], [13], [17], [23], [26], [30] assumes that the visual words in an image are generated by underlying, latent, topics. These topics, in turn, characterize a distribution over the visual word vocabulary. With the assumption that similar visual words often co-occur, a generative model may deal with *visual word uncertainty* since similar visual words will be modeled by the same topic. Moreover, a generative model may take *visual word plausibility* into account because non-representative visual words will attain low probabilities. A generative probabilistic model, however, is dependent on large amount of visual word co-occurrence counts, or co-occurrence with the same other words, to properly model ambiguity. In contrast, directly modeling visual word ambiguity does not rely on such constraints. What is more, since a generative visual word model builds on top of visual word assignments, direct ambiguity modeling can be used as input for a generative model. In this paper, we do not take generative models into account. A generative model on top of our ambiguity modeling would add another layer of complexity. This additional complexity complicates measuring the effect of direct ambiguity modeling. Since we are interested in measuring ambiguity, we concentrate on direct ambiguity modeling.

III. Visual Word Ambiguity by Kernel Codebooks

Given a vocabulary of codewords, the traditional codebook approach describes an image by a distribution over codewords. For each word w in the vocabulary V the traditional codebook model estimates the distribution of codewords in an image by

$$\mathrm{CB}(w) = \frac{1}{n}\sum_{i=1}^{n}\begin{cases} 1 & \text{if } w = \underset{v\in V}{\arg\min}(D(v, r_i)); \\ 0 & \text{otherwise,}\end{cases} \qquad (1)$$

where n is the number of regions in an image, r_i is image region i, and $D(w, r_i)$ is the distance between a codeword w and region r_i. Typically, the regions r_i are detected interest regions, or densely sampled image patches. The codebook model represents an image by a histogram of word frequencies that describes the probability density over codewords.

A robust alternative to histograms for estimating a probability density function is kernel density estimation [3], [36]. Kernel density estimation uses a kernel function to smooth the local neighborhood of data samples. A one-dimensional estimator with kernel K and smoothing parameter σ is given by $\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} K_\sigma(x - X_i)$, where n is the total number of samples and X_i is the value of sample i.

Kernel density estimation makes use of a kernel with a given shape and size. The kernel size determines the amount of smoothing between data samples whereas the shape of the kernel is related to the distance function between data samples [33], [37]. In this paper we use the SIFT descriptor that draws on the Euclidian distance as its distance function [38]. The Euclidean distance assumes a Gaussian distribution of the SIFT features, with identity as the covariance. Hence, the Euclidian distance is paired with a Gaussian-shaped kernel $K_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{1}{2}\frac{x^2}{\sigma^2})$. The

	Best Candidate	**Multiple Candidates**
Constant Weight	Traditional Codebook	Codeword Uncertainty
Kernel Weighted	Codeword Plausibility	Kernel Codebook

TABLE I
THE RELATIONSHIP BETWEEN VARIOUS FORMS OF CODEWORD AMBIGUITY AND THEIR PROPERTIES.

Gaussian kernel assumes that the variation between an image feature and a codeword is described by a normal distribution. This normal distribution has a smoothing parameter σ which represents the size of the kernel. This smoothing parameter determines the degree of similarity between data samples, and is dependent on the dataset, the feature dimensionality, and the range of the feature values. Note that we do not try to obtain the best fit of the data. In contrast, we aim to find the kernel size that discriminates best between classes. Therefore, we tune the kernel size discriminatively by cross-validation. Hence, the size of the kernel is dependent on the dataset and the image descriptor, whereas the shape of the kernel follows directly from the distance function.

In the codebook model, the histogram estimator of the codewords may be replaced by a kernel density estimator. Moreover, a suitable kernel (such as the Gaussian kernel) allows kernel density estimation to become part of the codewords, rather than the data samples. Specifically, when the kernel is symmetric, $K_\sigma(x - X_i) = K_\sigma(X_i - x)$, it trivially follows that there is no distinction between placing the kernel on the data sample or placing the kernel on a codeword. That is, if the centre of the kernel coincides with the codeword position, the kernel value at the data sample represents the same probability as if the centre of the kernel coincides with the data sample. Hence, a symmetric kernel allows for transferring the kernel from the data samples to the codewords, yielding a *kernel codebook*,

$$\text{KCB}(w) = \frac{1}{n}\sum_{i=1}^{n} K_\sigma\left(D(w, r_i)\right), \tag{2}$$

where n is the number of regions in an image, r_i is image region i, $D(w, r_i)$ is the distance between a codeword w and region r_i, and σ is the smoothing parameter of kernel K. The outcome now is represented by a continuous variable rather than a discrete one.

In essence, a kernel codebook alleviates the hard mapping of features in an image region to the codeword vocabulary. This soft-assignment models two types of ambiguity between codewords: codeword uncertainty and codeword plausibility. Codeword uncertainty indicates that one image region may distribute probability mass to more than one codeword. Conversely, codeword plausibility signifies that an image feature may not be close enough to warrant representation by any relevant codeword in the vocabulary. Each of these two types of codeword ambiguity may be modeled individually. *Codeword uncertainty*,

$$\text{UNC}(w) = \frac{1}{n}\sum_{i=1}^{n} \frac{K_\sigma\left(D\left(w, r_i\right)\right)}{\sum_{j=1}^{|V|} K_\sigma\left(D(v_j, r_i)\right)}, \tag{3}$$

normalizes the amount of probability mass to a total constant weight of 1 and is distributed over all relevant codewords. Relevancy is determined by the ratio of the kernel values for all codewords v in the vocabulary V. Thus, codeword uncertainty retains the ability to select multiple candidates, however does not take the plausibility of a codeword into account. In contrast, *codeword plausibility*,

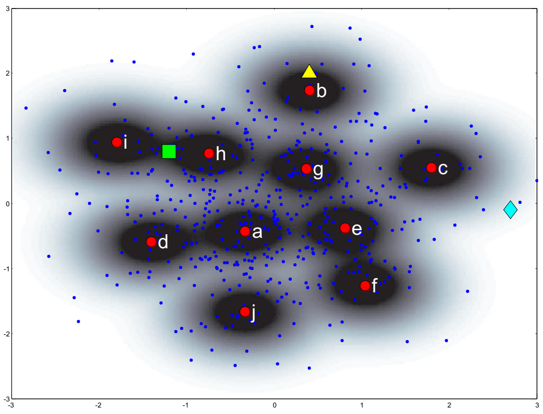


Fig. 2. An example of the weight distribution of a kernel codebook with a Gaussian kernel, where the square, diamond and triangle represent the image features taken from figure 1.

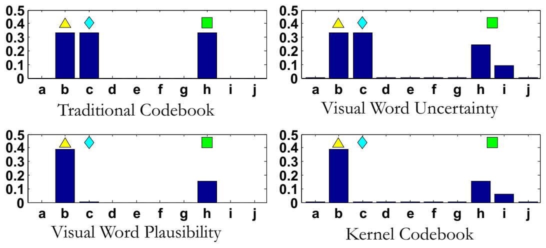


Fig. 3. Summary of different types of codeword ambiguity, according to table I. These distributions are based on the kernels shown in figure 2.

$$\text{PLA}(w) = \frac{1}{n}\sum_{i=1}^{n} \begin{cases} K_\sigma\left(D(w, r_i)\right) & \text{if } w = \arg\min_{v \in V}(D(v, r_i)); \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

selects for an image region r_i the best fitting codeword w and assigns it probability mass proportional to the kernel value of that codeword. Hence, codeword plausibility will give a higher weight to more relevant data samples. However, it cannot select multiple codeword candidates. Note that the selection of a single codeword retains sparsity, which is advantageous for large datasets. The relation between codeword plausibility, codeword uncertainty, the kernel codebook model, and the traditional codebook model is summarized in table I.

An example of the weight distributions of the various types of codeword ambiguity with a Gaussian kernel is shown in figure 2. Furthermore, in figure 3 we show an example of various codeword distributions corresponding to different types of codeword ambiguity. Note the weight difference in codewords for the data samples represented by the diamond and the square. Where the diamond contributes full weight in the traditional codebook, it barely adds any weight in the kernel codebook and codeword plausibility model. This may be advantageous, since it incorporates the implausibility of outliers. Furthermore, in the traditional codebook, the square adds weight to one single codeword, whereas the kernel codebook and codeword uncertainty adds weight to the two relevant codewords. In the latter two methods, the uncertainty between the two codewords is not assigned solely to the best fitting word, but divided over both

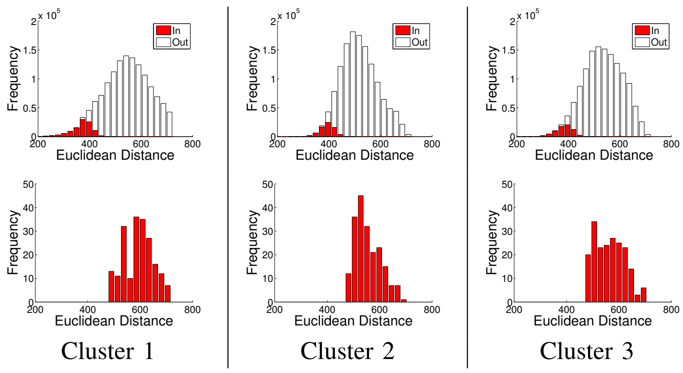


Fig. 4. Histograms of Euclidean distances over 200 clusters, where each column represents a different cluster. The top row displays the distances from this cluster to all other points in the train set. The data samples that are closest to this cluster are indicated as *in* whereas the points that are assigned to other clusters are denoted *out*. The bottom row shows the distances from this cluster center to all other cluster centers.

codewords. Hence, the kernel codebook approach can be used to introduce various forms of ambiguity in the tradition codebook model. We will experimentally investigate the effects of all forms of codeword ambiguity in section IV.

In this paper we consider the kernel size fixed for all codewords. We have considered a variable kernel density estimator [36], where the smoothing factor σ varies per codeword. This variable smoothing factor could be determined by the variance of the image features that are assigned to each codeword by the clustering algorithm. However, varying the kernel size for each codeword yields an inhomogeneous feature space where distances are measured differently depending on their location in feature space. Essentially, a varying kernel size makes certain codewords more important than others. This difference in codeword importance may be justified, however, should be tied to the final classification performance. Since the classification performance is not taken into account by an unsupervised clustering algorithm, we adhere to a homogenous feature space by keeping the kernel size fixed for all codewords.

The ambiguity between codewords will be influenced by the number of words in the vocabulary. A large vocabulary allows a rich selection of visual words, increasing the likelihood that an image feature is well-represented. Moreover, in the case of a large vocabulary, the probability of multiple relevant visual words increases, suggesting the use of visual word uncertainty. On the other hand, when the vocabulary is small, essentially different image parts will be represented by the same vocabulary element. This misrepresentation may be alleviated by considering visual word plausibility. Hence, visual word ambiguity influences both small and large vocabularies. We extensively investigate the effect of the vocabulary size in section IV.

Since codewords are image descriptors in a high-dimensional feature space, we expect a relationship between codeword ambiguity and feature dimensionality. With a high-dimensional image descriptor, codeword ambiguity will become more significant. If we consider a codeword as a high-dimensional sphere in feature space, then most feature points in this sphere will lay on a thin shell near the surface. Hence, in a high-dimensional space, more feature points will be close to the boundary between codewords than in a lower-dimensional feature space. Thus, they introduce ambiguity between codewords. See Bishop's textbook on pattern recognition and machine learning [33, Chapter 1, pages 33–38] for a thorough explanation and illustration of the curse of dimensionality. Consequently, increasing the dimensionality of the image descriptor will in general increase the level of codeword ambiguity. In the next section we will experimentally investigate the effects of the dimensionality of the image descriptor.

IV. Experiments

We experimentally compare codeword ambiguity modeling against the traditional codebook approach for five large and varied datasets: fifteen natural scene categories from Lazebnik *et al.* [14], Caltech-101 by Fei-Fei and Perona [39], Caltech-256 by Griffin *et al.* [40] and the Pascal VOC sets of 2007 [41] and 2008 [42]. We start our experiments with an in-depth analysis of our methods on the set of fifteen natural scene categories, after which we transpose these findings to the experiments on the two Caltech sets and the two issues of Pascal VOC. For our experimental setup we closely follow Lazebnik *et al.* [14] as this setup has shown excellent performance on these datasets.

A. Experimental Setup

To obtain reliable results, we repeat the experimental process 10 times. We select 10 random subsets from the data to create 10 pairs of train and test data. For each of these pairs we create a codeword vocabulary on the train set. The exact same codeword vocabulary is used by both the codebook and the codeword ambiguity approaches to describe the train and the test set. For classification, we use an SVM with a histogram intersection kernel. Specifically, we use libSVM, and use the built in one-versus-one approach for multi-class classification. We use 10-fold cross-validation on the train set to tune parameters of the SVM and the size K_σ of the codebook kernel. The classification rate we report is the average of the per-category recognition rates which in turn are averaged over the 10 random test sets.

For image features we follow Lazebnik *et al.* [14], and use a SIFT descriptor sampled on a regular grid. A grid has been shown to outperform interest point detectors in image classification [8], [12], [20]. We compute all SIFT descriptors on overlapping 16x16 pixel patches, computed over a dense grid sampled every 8 pixels. Due to small implementation differences, our re-implementation of [14] performs slightly under their reported results. However, we use the same re-implementation for all methods of codeword ambiguity. Thus we do not bias any method by a slightly different implementation.

We create a codeword vocabulary by radius-based clustering. Radius-based clustering ensures an even distribution of codewords over the feature space and has been shown to outperform the popular k-means algorithm [12]. Whereas Jurie and Triggs [12] use mean-shift with a Gaussian kernel to find the densest-point, we maximize the number of data samples within its radius r for efficiency reasons.

In figure 4 we illustrate for 200 clusters the effect of clustering. We show the similarity distribution from cluster centers to other SIFT descriptors. The similarity distribution adheres to a Weibull shape, as expected [44]. For the Scene-15 dataset, the radius-based clustering algorithm used a radius of $r = 240$ to arrive at 200 clusters. Note, that this radius guarantees that the next cluster is at least a distance of $2r$ away, as can be seen in the bottom row of figure 4. Furthermore, note that for each cluster, the distance distribution from the cluster to all points (top row)

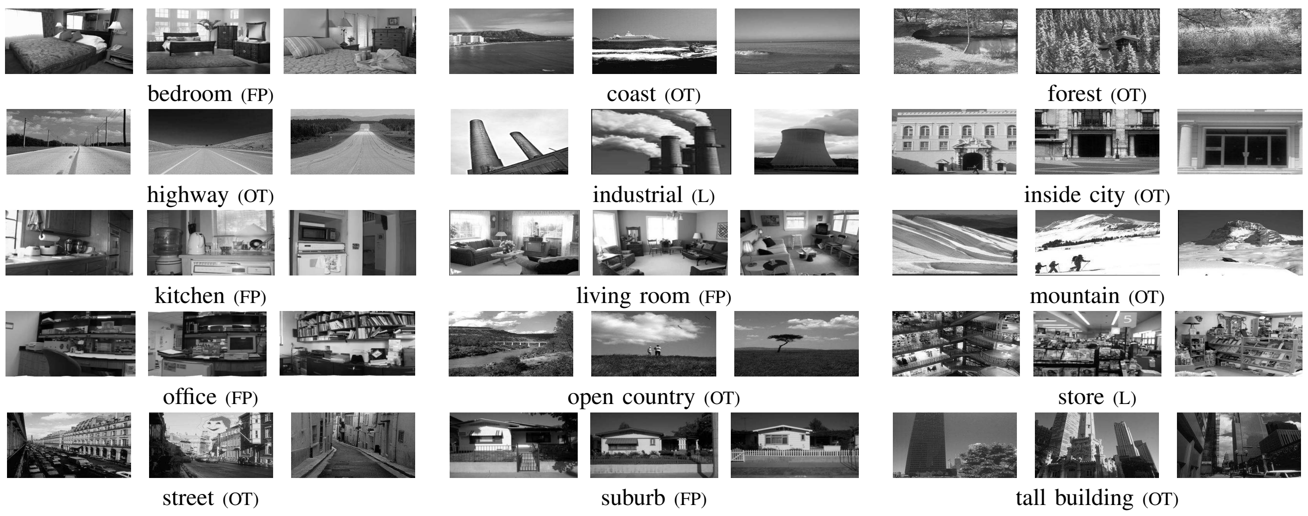


Fig. 5. Example images from the Scene-15 dataset. Each category is labeled with the annotator, where (OT) denotes Oliva and Torralba [43], (FP) is Fei-Fei and Perona [8], and (L) refers to Lazebnik *et al.* [14].

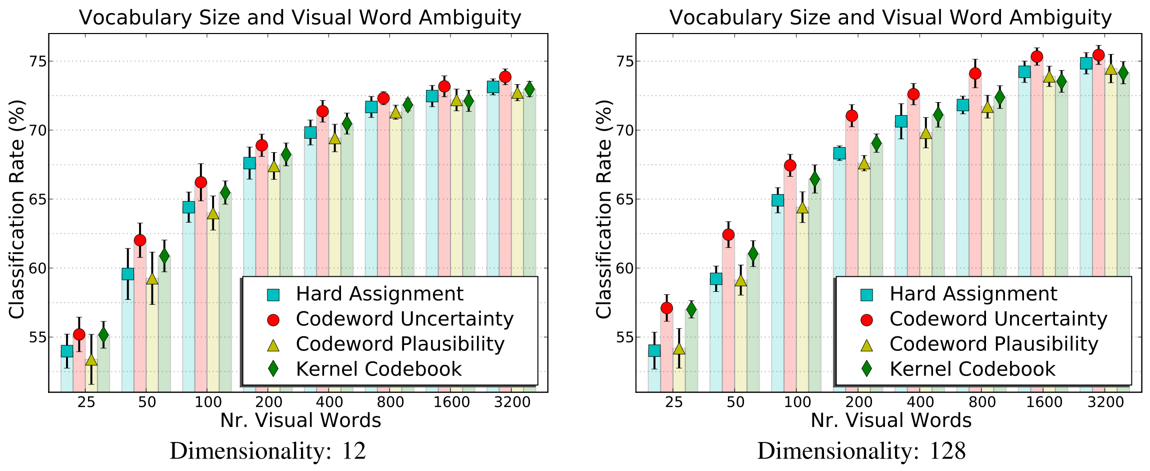


Fig. 6. Classification performance results of various types of codeword ambiguity for the Scene-15 dataset over various vocabulary sizes and feature dimensions.

is fairly similar to the distance distribution from this cluster to the other clusters (bottom row). This similarity suggests that the clusters give a good representation of the complete data.

B. *Experiment 1: In-depth Analysis on the Scene-15 Dataset*

The first dataset we consider is the Scene-15 dataset, which is compiled by several researchers [8], [14], [43]. The Scene-15 dataset consists of 4,485 images spread over 15 categories. The fifteen scene categories contain 200 to 400 images each and range from natural scenes like mountains and forests to man-made environments like kitchens and offices. In figure 5 we show examples of the scene dataset. We use an identical experimental setup as Lazebnik *et al.* [14], and select 100 random images per category as a train set and the remaining images as the test set.

For the Scene-15 dataset, we analyze the types of codeword ambiguity, vocabulary size and feature dimensionality. To evaluate the effect of feature dimensionality on visual word ambiguity we project the 128 length SIFT descriptor to a lower dimensionality. This dimension reduction is achieved with principal component analysis, which reduces dimensionality by projecting the data on a reduced-dimensional basis while retaining the highest variance in the data. We compute a reduced basis on each complete training set, after which we project the train set and corresponding test set on this basis. We reduce the feature length from 128 dimensions to 12 dimensions. A projection to 60 dimensions shows very similar results (data not shown). In evaluating vocabulary size, we tune the radius in the radius-based clustering algorithm to construct eight differently sized vocabularies. The vocabulary sizes we consider are $\{25, 50, 100, 200, 400, 800, 1600, 3200\}$. The results for all types of codeword ambiguity evaluated for various vocabulary sizes and the two feature dimensionalities (12 and 128) are given in figure 6.

We start the analysis of the results in figure 6 with the various types of codeword ambiguity. The results show that codeword uncertainty consistently outperforms other types of ambiguity for all dimensions and all vocabulary sizes. This performance gain is not always significant, however. Nevertheless, for 128 dimensions and a vocabulary size of 200, codeword uncertainty (UNC) outperforms hard assignment with a vocabulary size of 400 and this trend holds for larger vocabulary size pairs: (200-UNC $>$ 400-HARD), (400-UNC $>$ 800-HARD), (800-UNC $\geq$ 1600-UNC) and (1600-UNC $>$ 3200-HARD). On the other end of the performance scale there is codeword plausibility, which always yields the worst results. The third option, a kernel codebook, outperforms hard

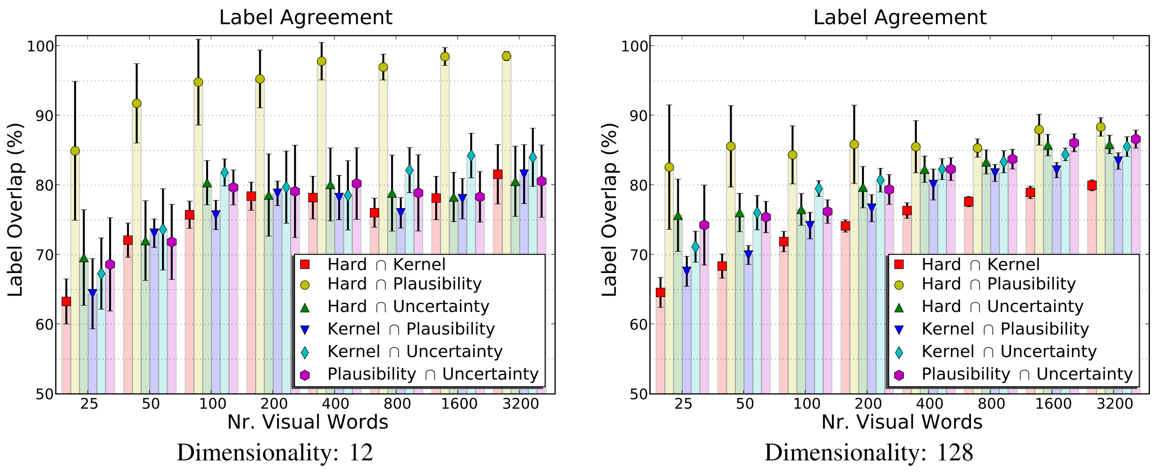


Fig. 7. Analysis of the class label overlap as predicted by various types of codeword ambiguity for the Scene-15 dataset.

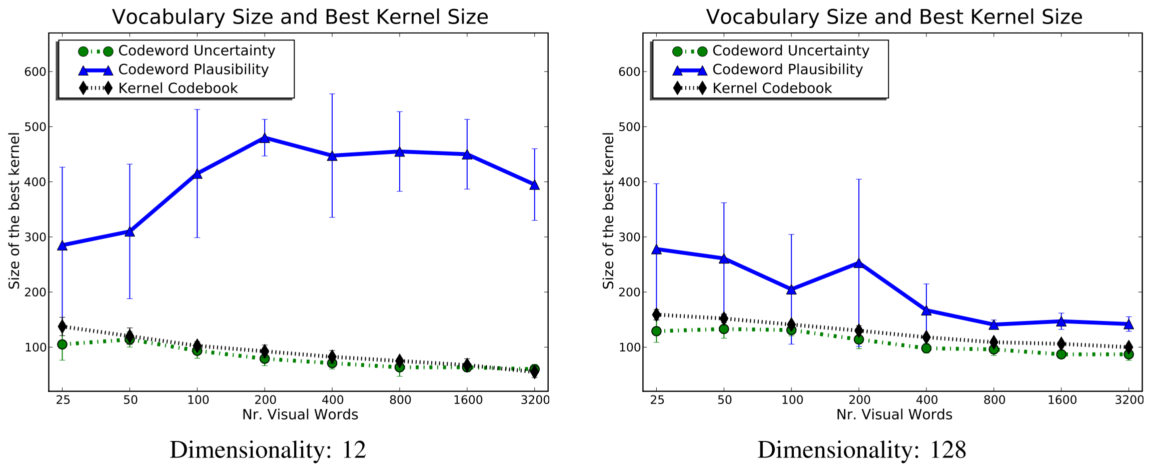


Fig. 8. Analysis of the best kernel size, found with 10-fold cross-validation, used by various types of codeword ambiguity for the Scene-15 dataset.

assignment for smaller vocabulary sizes. For smaller vocabulary sizes the differences between codeword ambiguity types become more pronounced, whereas using a larger vocabulary dampens the differences between ambiguity types. And, as expected, the highest performance gain for codeword ambiguity is in a higher-dimensional feature space. When taking overall performance into account, the results indicate that a higher-dimensional descriptor yields the best results. Moreover, increasing the vocabulary size seems to asymptotically improve performance for all methods. We will investigate larger vocabularies in more detail, later.

To gain insight in the performance variation between the various types of codeword ambiguity we show the overlap percentage between the predicted category labels for all paired method in figures 7. The first thing that is striking in figure 7, is the high category label overlap between hard assignment and codeword plausibility. This high overlap may be explained by noting that codeword plausibility resembles hard assignment when the kernel size is sufficiently large. Inspecting the kernel sizes as found with cross-validation reveals that the kernel size for codeword plausibility is indeed large. The kernel size for codeword plausibility is typically 200 or larger, whereas the other types of codeword ambiguity range around 100. Furthermore, this label overlap between hard assignment and codeword plausibility is highest with a small number of dimensions. This may be due to the fact that a higher-dimensional space leaves more room for implausible features than a lower dimensional space. The kernel codebook and hard assignment pair have the least number of labels in common. This low label overlap may be expected, since these two types represent the extremes of the types of codeword ambiguity as shown in table I. Further differences of label overlap can be seen between the low- and the high-dimensional feature space. In a high-dimensional feature space there tends to be less correlation between category labels. In a high-dimensional space, the differences between the types of ambiguity become more pronounced, reducing the label overlap. A further trend may be observed in the increased overlap for an increasing vocabulary size. Increasing the vocabulary size yields an increased performance, which requires more labels to be predicted correctly. We attribute the increase in label overlap to those images that are predicted correctly by a larger vocabulary. This link between increased performance and increased category label overlap also explains that the category label overlap is generally high between all types of codeword ambiguity.

To evaluate the influence of the kernel size, we show the kernel size found with 10-fold cross-validation in figure 8. The figure shows the optimal kernel size for the various ambiguity types for the two feature dimensions and for an increasing vocabulary size. The kernel size for codeword uncertainty and the kernel codebook show a low variance over the 10 random repetitions. This indicates that these two types of codeword ambiguity have

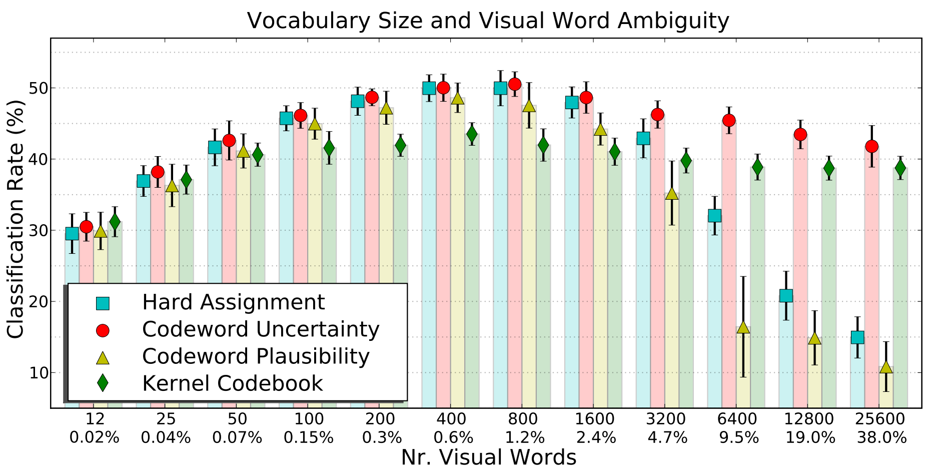


Fig. 9. Classification performance results of various types of codeword ambiguity for the Scene-15 dataset, trained on 5 images per class. This figures illustrates the effect of relatively large vocabulary sizes compared to the total number of image features.

a stable, optimal kernel size. In contrast, the best kernel sizes for codeword plausibility fluctuate heavily over the 10 repetitions. Analyzing the scores, we found that increasing the kernel size of codeword plausibility beyond a sufficiently large value does not change the classification scores much. *I.e.*, for large kernel sizes there are no implausible features left in the finite feature space. Therefore, sufficiently large kernels lead to similar classification performance without a clear optimum, resulting in high kernel size variance for codeword plausibility. In analyzing the kernel size over the number of vocabulary elements shows that a larger vocabulary leads to slightly smaller kernels. This may be expected, since a larger vocabulary is formed by a smaller radius between codewords. When considering the dimensionality of the descriptor, it shows that lower dimensional features use a smaller kernel. This is the case because low-dimensional features typically have a smaller Euclidean distance than high-dimensional features. In summary, the kernel size depends on the type of ambiguity, feature dimensionality and the number of codewords. Therefore, the optimal kernel size cannot be easily inferred from the data and should be found in a discriminative manner, linking it directly to classification performance as achieved with cross-validation.

As illustrated in figure 6, increasing the vocabulary size increases the classification performance and the performance of the four ambiguity types seems to converge. In figure 6, however, the vocabulary sizes are relatively small. The largest vocabulary in figure 6 has 3200 elements and comprises only 0.23% of all features. The behavior of relatively small vocabularies may not be identical to relatively large vocabularies. With vocabulary sizes that are relatively large compared to the total number of training image features, ambiguity type performance may diverge again. To evaluate this, we compared ambiguity type performance on the Scene-15 dataset over relatively large vocabularies.

To make the computation of relatively large vocabularies practically feasible, we reduced the total number of features in the training set. The number of features may be reduced by only extracting features on detected interest points in an image. However, interest point detection would deviate too much from our uniform experimental setup for the Scene-15 dataset. Hence, we keep extracting image features on a regular grid yet constrain the total number of image features by reducing the number of images per class as is also done by [4], [8], [40]. For this experiment, we randomly select 5 images for each of the 15 classes, using the remaining images for the test set. The average number of training feature over the 10 random repetitions amounts to a total of $67,408 \pm 348$ unique SIFT descriptors. Our experiment is not as much concerned with the total number of features per se, but with the ratio between the number of features and the size of the vocabulary. We want to measure the effect of *relatively* large vocabularies. We evaluated vocabulary sizes ranging from 12 (0.02%) to 25,600 (38%) unique visual words. The underlying assumption is that the results in this experiment trend will hold for various feature and vocabulary sizes, however with similar ratios.

The results for relatively large vocabularies are given in figure 9. Note that the performance for relatively small vocabularies show a similar trend as in figure 6. Hence, the results in figure 6 and figure 9 are in agreement. The main difference is the lower performance in figure 9 because only 5 images per class are used for training. In figure 9 it can be seen that for vocabulary sizes larger than 800 visual words (1.2%), the performance of all methods decreases. We attribute this performance decrease to the curse of dimensionality, albeit that we use a discriminative SVM classifier. In analyzing ambiguity types, it can be seen that for vocabulary sizes of 6,400 and higher, the performance of hard assignment and visual word plausibility severely deteriorates. This may be expected, since both of these ambiguity types can not select multiple suitable visual words. For example, in the extreme case of a vocabulary size equal to the number of image features, codeword plausibility and hard assignment map each training image feature to it's own unique visual word, reverting to exact feature matching. In contrast, the kernel codebook and codeword uncertainty methods both allow selecting multiple relevant visual words. When increasing the vocabulary size, the performance of these two types remains relatively stable, where codeword uncertainty is the better performer. As shown by this experiment, a larger vocabulary does not necessarily yield better results. Actually, a too large vocabulary severely deteriorates performance for codeword plausibility and hard-assignment. A kernel codebook and codeword uncertainty, however, only decrease slightly. Hence, for relatively large vocabularies visual word ambiguity modeling makes a significant difference.

To show the modularity of our approach and improve results

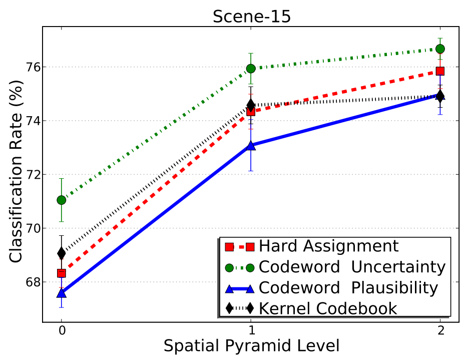


Fig. 10. Classification performance on the Scene-15 dataset of various types of codeword ambiguity using the spatial pyramid.

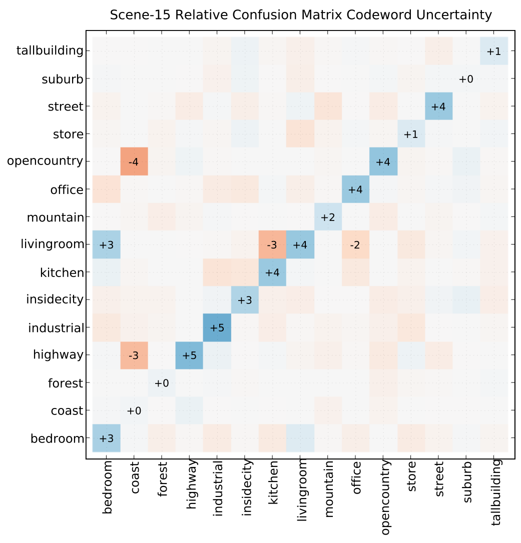


Fig. 11. Relative confusion matrix of the Scene-15 dataset, for 200 codewords at level 0 of the pyramid, best viewed in color. The relative confusion denotes the increase (blue) or decrease (red) of the absolute classification score of codeword uncertainty compared to hard assignment matrix. We show the average classification percentage per category. The value at column x and row y represents the difference between codeword uncertainty and hard assignment in classifying images of category y as category x.

we incorporate the spatial pyramid by Lazebnik *et al.* [14]. The spatial pyramid divides an image into a multi-level pyramid of increasingly fine subregions and computes a codebook descriptor for each subregion. The spatial pyramid has been shown to yield excellent performance [5], [14], [15]. We use the 128 dimensional features and a vocabulary of 200 codewords in accordance with Lazebnik *et al.* [14]. The results for the various forms of codeword ambiguity for the first two levels of the spatial pyramid are shown in figure 10. Our best result with codeword uncertainty is $76.7 \pm 0.4\%$, whereas hard assignment scores $75.8\pm$ 0.6%, both on level 2 of the pyramid. Codeword uncertainty at pyramid level 1 outperforms the traditional codebook at pyramid level 2, effectively saving a complete pyramid level. For the Scene-15 dataset, codeword uncertainty gives the highest improvement at level 0 of the spatial pyramid, which is identical to a codebook model without any spatial structure. Nevertheless, codeword uncertainty outperforms the hard assignment of the traditional codebook for all levels in the pyramid.

The relative confusion matrix of the Scene-15 dataset for 200

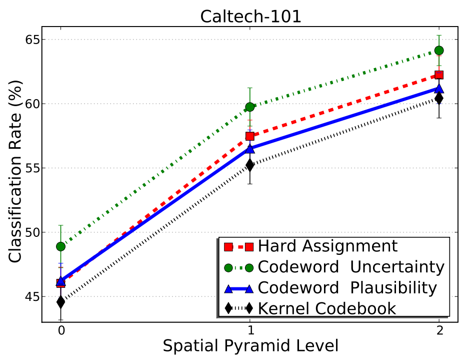


Fig. 12. Classification performance on the Caltech-101 dataset of various types of codeword ambiguity using the spatial pyramid.

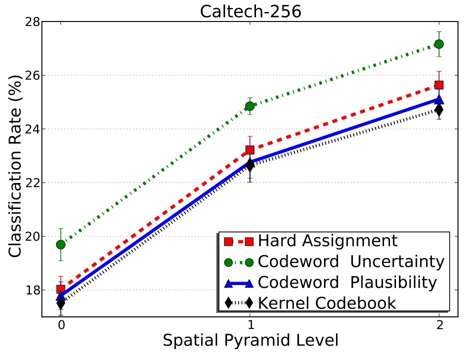


Fig. 13. Classification performance on the Caltech-256 dataset of various types of codeword ambiguity using the spatial pyramid.

codewords at level 0 of the pyramid is shown in figure 11. The relative confusion denotes the absolute difference between entries in the confusion matrix of codeword uncertainty relative to the matrix of hard assignment. We focus on hard assignment versus codeword uncertainty, since uncertainty gives the highest improvement of the three types of visual word ambiguity. The non-diagonal entries that represent misclassification rates mostly decrease, or do not change much. The only pair with a higher confusion rate is the confusion between *livingroom* as *bedroom*. Nevertheless, this confusion is compensated by increased discriminative ability between *livingroom* and the categories *kitchen* and *office*. Further considerable confusion reduction is between *open country* as *coast* and *highway* as *coast*. Note that codeword uncertainty improves or matches the correct classification performance for all categories, given by the diagonal.

C. Experiment 2 and 3: Caltech-101 and Caltech-256

We conduct our second set of experiments on the Caltech-101 [39] and Caltech-256 [40] datasets. The Caltech-101 dataset contains 8,677 images, divided into 101 object categories, where the number of images in each category varies from 31 to 800 images. The Caltech-101 is a diverse dataset, however the obects are all centered, and artificially rotated to a common position. In figure 14 we show some example images of the Caltech-101 set. Some of the problems of Caltech-101 are solved by the Caltech-256 dataset. The Caltech-256 dataset holds 29,780 images in 256 categories where each category contains at least 80 images. The Caltech-256 dataset is still focused on single objects. However, in contrast to the Caltech-101 set, each image is not manually rotated

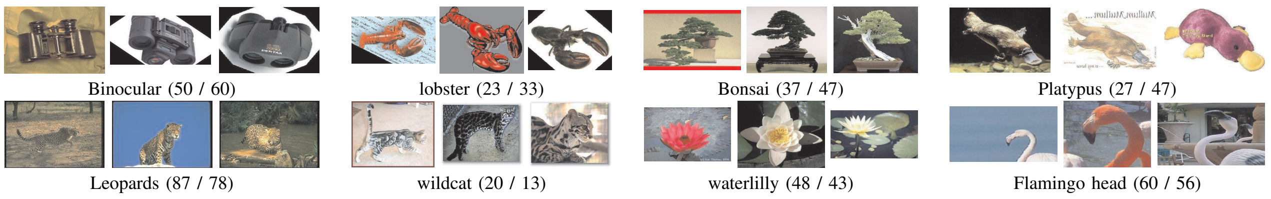


Fig. 14. Examples of the Caltech-101 set. Top: the top 4 categories where our method improves most, Bottom: the 4 categories where our method decreases performance. The numbers in brackets indicate the classification rate (hard / uncertainty).

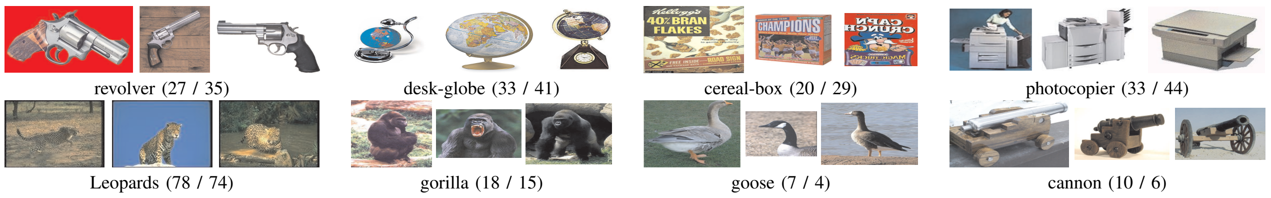


Fig. 15. Examples of the Caltech-256 set. Top: the top 4 categories where our method improves most, Bottom: the 4 categories where our method decreases performance most. The numbers in brackets indicate the classification rate (hard / uncertainty).

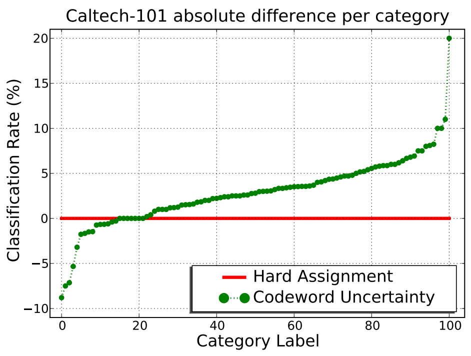


Fig. 16. The classification performance difference per category between hard assignment and codeword uncertainty for Caltech-101.

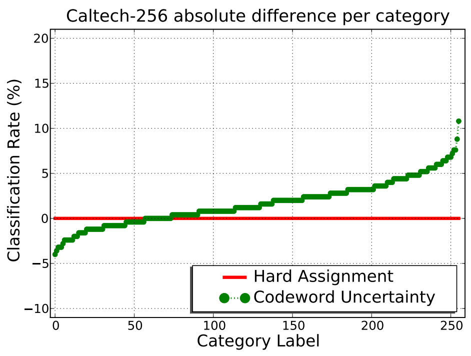


Fig. 17. The classification performance difference per category between hard assignment and codeword uncertainty for Caltech-256.

to face one direction. In figure 15 we show some example images of the Caltech-256 set. We report classification performance on both Caltech sets.

Our experimental results for both the Caltech-101 as Caltech-256 are generated by 30 images per category for training. For testing, we employed 50 images per category for the Caltech 101, and 25 images per category for the Caltech-256. These number of train and test images are typically used for these sets [14], [40]. We use 128 dimensions, and compare the four types of visual word ambiguity. The average classification results per spatial pyramid level for Caltech-101 and Caltech-256 are shown in figure 12 and figure 13. These results on Caltech are similar to the results on the Scene-15 dataset. For both sets, the codeword uncertainty method outperforms the traditional codebook considerably in the light of the difficulty of the problem and the simplicity of the improvement. Our best result for Caltech-101 with codeword uncertainty is $64.1 \pm 1.5\%$, whereas hard assignment scores $62.2\pm$ 1.2%, both on level 2 of the pyramid. For Caltech-256 our best result is $27.2\pm$ 0.4%, whereas hard assignment scores $25.63\pm$ 0.5%. The classification performance difference per category between hard assignment and codeword uncertainty are given in figure 16 and figure 17. For the Caltech-101 set, there are 86 categories that perform better, or equal with codeword uncertainty. In the case of the Caltech-256 set, there are 199 categories with better or equal performance.

The relative confusion matrices of each Caltech dataset for

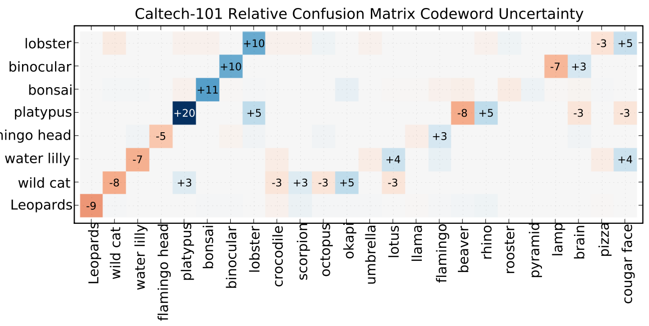


Fig. 18. Relative confusion matrix for the Caltech-101 dataset, best viewed in color. We show the 4 categories that increase most, and the 4 categories that decrease performance most. Each of these 8 categories is paired with its most confusing and least confusing category. The value at column x and row y represents the difference between codeword uncertainty and hard assignment in classifying images of category y as category x.

200 codewords at level 0 of the pyramid are given in figure 18 and figure 19. The relative confusion denotes the difference between entries in the confusion matrix of codeword uncertainty compared to the matrix of hard assignment. Since the size of these datasets prohibits displaying the full confusion matrix, we show the four categories that increase most, and the four categories that decrease most by using codeword uncertainty over hard assignment. Moreover, for each of these categories we show their

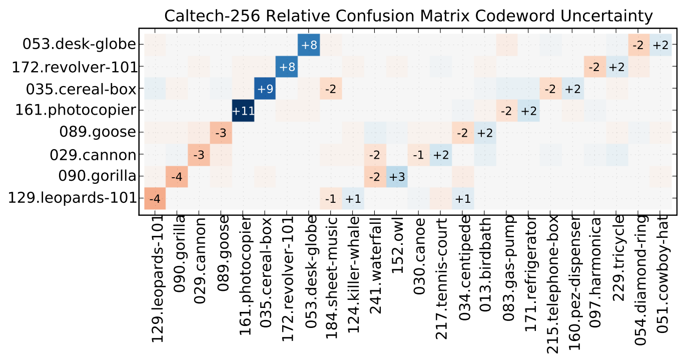


Fig. 19. Relative confusion matrix for the Caltech-256 dataset, for the top 4 and bottom 4 categories as in figure 18. The value at column x and row y represents the difference between codeword uncertainty and hard assignment in classifying images of category y as category x.

most confusing, and least confusing category. We focus on the difference between codeword uncertainty and hard assignment, since the former gives the best results and the latter is most commonly used in literature. Some examples of the classes that increase, and decrease most are given in figures 14 and 15.

We start the analyses of the relative confusing matrices of the Caltech datasets with the categories where performance decreases most. These object categories consist mostly of natural images that are captured including their contextual background. We deem this background as the reason for a decreased performance by ambiguity modeling. The background is very similar for several natural images. By incorporating ambiguity modeling this similarity is enhanced, leading to more confusion. In analyzing the categories that improve most, we observe that these categories mainly consist of man-made objects, and objects that are photographed without context. We conjecture that the reason why these classes benefit most from codeword ambiguity is that these object classes have little intra-class variation. Small variations may lead to completely different codewords when using the hard assignment as in the traditional codebook model. In contrast, our approach of ambiguity modeling will reserve weight for multiple, suitable codewords, leading to classification improvements.

D. Experiment 4: PASCAL VOC07-20 and VOC08-20 Datasets

As a final experiment, we consider the Pascal VOC 2007 [41] and 2008 challenge [42]. The VOC challenges consist of twenty object classes with 9,963 images in 2007 and in 10,057 images in 2008. These image sets are each split in half to a given train and test set. The Pascal VOC Challenge provides a yearly benchmark of object recognition algorithms. We follow the successful approach by Marszałek *et al.* [16], which was extended by Tahir and Van de Sande *et al.* [27]. Specifically, for each image we combine Harris-Laplace point sampling with densely sampling every 6 pixels. These points are subsequently represented by SIFT, and various color-SIFT descriptors [24]. The descriptors of the train set with around 5000 features per image are subsequently clustered by k-means to create a vocabulary of 4,000 codewords. This vocabulary is used in the codebook model at level 1 of Lazebniks spatial pyramid where we use a support vector machine with a χ^2 kernel for image classification. We fuse the classification scores for the various SIFT descriptors with a simple geometric mean. The final classification performance is measured in average precision, which represents the area under the precision-recall graph.

We experimentally compared the traditional codebook model with codeword uncertainty and with the best two participating systems on the respective Pascal challenge. In figure 20 we show the results for both Pascal VOC 2007 and 2008. For Pascal VOC 2007 (VOC07-20) our implementation with codeword uncertainty performs best for 15 of the 20 object classes. The best method for the 5 other object classes is INRIA_Genetic. In terms of mean average precision over all object classes, our implementation with codeword uncertainty scores best with 0.605, followed by INRIA_Genetic with 0.594, hard assignment with 0.580 and XRCE with 0.556. Note that the traditional codebook model occupies the third place, whereas replacing hard assignment with codeword uncertainty yields the best result. Moreover, codeword uncertainty outperforms hard assignment for all 20 categories of VOC07-20. In the case of Pascal VOC 2008 (VOC08-20), SurreyUvA_SRKDA claims 9 categories, LEAR_shotgun wins 9, and codeword uncertainty is the best for 4 categories[1]. The best system in mean average precision is SurreyUvA_SRKDA with 0.549, followed by LEAR_shotgun with 0.545, codeword uncertainty with 0.541 and 0.521 for the traditional codebook. The SurreyUvA_SRKDA system with the best mean average precision already uses codeword uncertainty as a part of their method [27]. The main difference between SurreyUvA_SRKDA and our results presented here, is the use of a classifier with multiple kernel learning which is out of scope for this article. In comparing hard assignment with codeword uncertainty, the latter slightly decreases the performance for the category *cow*. For the other 19 categories of VOC08-20 the performance of codeword uncertainty is equal or better than hard assignment.

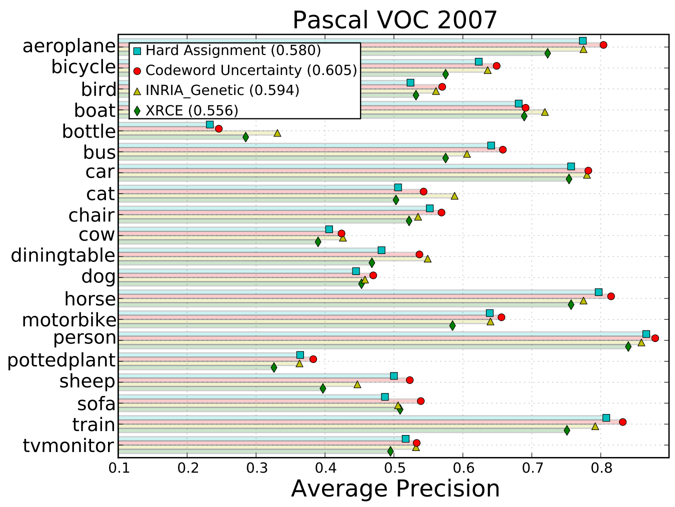
V. Discussion

This paper presented a principal improvement on the popular codebook model for scene classification. The traditional codebook model uses hard assignment to represent image features with codewords. We replaced this basic property of the codebook approach by introducing uncertainty modeling, which is appropriate as discrete feature vectors are only capable of capturing part of the intrinsic variation in visual appearance. This uncertainty is achieved with techniques based on kernel density estimation.

The experiments on the Scene-15 dataset in figures 6 and 10 show that of the four considered ambiguity types, codeword plausibility hurts performance. Codeword plausibility (PLA), and the unnormalized kernel-codebook (KCB), are dominated by those few representative image features that are significantly close to a codeword. In essence, PLA and to a lesser extent KCB, ignore the majority of the features, and leads us to conclude that it is better to have an implausible codeword representing an image feature then no codeword at all. When no codeword is selected, all statistical classification techniques developed to deal with noisy data are not used to their full potential. Therefore, codeword uncertainty yields the best results, since it models ambiguity between codewords, without taking codeword plausibility into account.

The results in figure 6 indicate that codeword ambiguity is more effective for higher-dimensional features than for lower dimensions. The curse of dimensionality prophesizes that increasing the

[1]This totals to 22 because 2 systems share the best score for the categories *person* and *bus*.



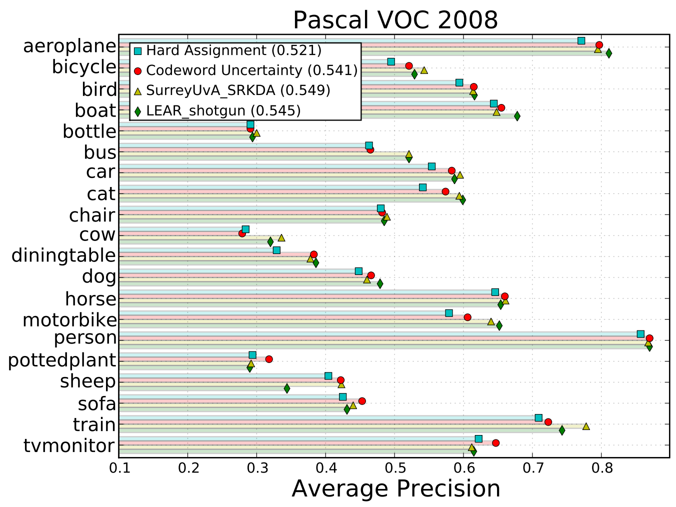


Fig. 20. Average Precision of the traditional codebook model and codeword uncertainty per category, compared against the best two participants for Pascal VOC 2007 (left) and Pascal VOC 2008 (right). The mean average precision for each method is shown in the legend. Note that codeword uncertainty is used by the SurreyUvA_SRKDA method that participated in Pascal VOC 2008.

Data set	Train set size	Test set size	Performance Increase
Scene-15	1,500	2,985	4.0 $\pm$ 1.7 %
Caltech-101	3,030	5,050	6.3 $\pm$ 1.9 %
Caltech-256	7,680	6,400	9.3 $\pm$ 3.0 %
VOC07-20	5,011	4,952	4.3 %
VOC08-20	4,340	5717	3.8 %

TABLE II
THE RELATIONSHIP BETWEEN THE DATA SET SIZE AND THE RELATIVE PERFORMANCE OF CODEWORD UNCERTAINTY OVER HARD ASSIGNMENT FOR 200 CODEWORDS IN THE SCENE-15 AND CALTECH DATASETS AND 4,000 CODEWORDS FOR THE VOC07-20 AND VOC08-20 SETS.

dimensionality increases the fraction of feature vectors on or near the boundary of codewords. Hence, increasing the dimensionality will increase codeword uncertainty, leading to better results for ambiguity modelling with higher-dimensional features.

Figure 6 seems to suggest that a larger vocabulary is always better. Furthermore, the figure suggests that for larger vocabularies the performance of hard assignment and soft-assignment converges. Figure 9 illustrates that both these suggestions are not the case. Figure 9 shows that a too large vocabulary severely deteriorates the performance of hard assignment, whereas codeword ambiguity degrades only slightly. In the case of the VOC2007/2008 with around 5000 images in the training set with close to 5000 features per image, a vocabulary of 4000 words is rather small. Because of this relatively small vocabulary there is a significant improvement of soft-assignment over hard assignment. Even more performance improvement can be expected by choosing a much larger vocabulary. However, as shown in figures 6 and 9, the positive effect of a larger vocabulary size on the performance decreases logarithmically. Hence, it takes a vocabulary size of several orders of magnitude higher to obtain a significant improvement. Such larger vocabularies makes it practically infeasible to compute all (color) descriptors, spatial pyramid levels, and machine learning techniques. In contrast, ambiguity modeling provides increased performance at much lower computational costs.

The results over the Scene-15, Caltech-101, Caltech-256, and Pascal datasets are summarized in table II. This table shows the relative improvement of codeword uncertainty over hard assignment. Note that the result for the Pascal datasets is set apart, since it adheres to a different experimental setup. As can be seen in this table, the relative performance gain of ambiguity modeling increases as the number of scene categories grows. A growing number of scene categories requires a higher expressive power of the codebook model. Since the effects of ambiguity modeling increase with a growing number of categories, we conclude that ambiguity modeling is more expressive then the traditional codebook model. The results of all experiments show that codeword uncertainty outperforms the traditional hard assignment over all dimensions, all vocabulary sizes, and over all datasets.

We have demonstrated the viability of our approach by improving results on recent codebook methods. These results are shown on five well-known datasets, where our method consistently outperforms the traditional codebook model. We have shown that ambiguity modeling can obtain the same performance as hard assignment with a considerable smaller vocabulary. What is more, we found that hard assignment suffers more from the curse of dimensionality, whereas our ambiguity modeling approach reaps higher benefits in a high-dimensional feature space. Furthermore, the performance of hard assignment completely deteriorates when using relatively large vocabularies, while the proposed model performs consistently. Similarly, an increasing number of scene categories increases the effectiveness of our method. As future image features and datasets are expected to increase in size, our ambiguity modeling method is unambiguously likely to have more impact.

ACKNOWLEDGMENT

The authors like to thank Koen van de Sande, for his work on the Pascal VOC set. This work was supported by MultimediaN.

REFERENCES

[1] A. Agarwal and B. Triggs, “Multilevel image coding with hyperfeatures,” *Int. J. Comput. Vision*, vol. 78, no. 1, 2008.
[2] D. Batra, R. Sukthankar, and T. Chen, “Learning class-specific affinities for image labelling,” in *CVPR*, 2008.

[3] O. Boiman, E. Shechtman, and M. Irani, “In defense of nearest-neighbor based image classification,” in *CVPR*, 2008.

[4] A. Bosch, A. Zisserman, and X. Munoz, “Scene classification using a hybrid generative/discriminative approach,” *TPAMI*, vol. 30, no. 4, pp. 712–727, 2008.

[5] ——, “Image classification using random forests and ferns,” in *ICCV*, 2007.

[6] M. Boutell, J. Luo, and C. Brown, “Factor-graphs for region-based whole-scene classification,” in *CVPR SLAM Workshop*, 2006.

[7] O. Chum and A. Zisserman, “An exemplar model for learning object classes,” in *CVPR*, 2007.

[8] L. Fei-Fei and P. Perona, “A bayesian hierarchical model for learning natural scene categories,” in *CVPR*, 2005.

[9] J. van Gemert, J. Geusebroek, C. Veenman, and A. Smeulders, “Kernel codebooks for scene categorization,” in *ECCV*, 2008.

[10] J. van Gemert, J. Geusebroek, C. Veenman, C. Snoek, and A. Smeulders, “Robust scene categorization by learning image statistics in context,” in *CVPR SLAM Workshop*, 2006.

[11] Y.-G. Jiang, C.-W. Ngo, and J. Yang, “Towards optimal bag-of-features for object categorization and semantic video retrieval,” in *CIVR*, 2007.

[12] F. Jurie and B. Triggs, “Creating efficient codebooks for visual recognition.” in *ICCV*, 2005, pp. 604–610.

[13] D. Larlus and F. Jurie, “Latent mixture vocabularies for object categorization,” in *BMVC*, 2006.

[14] S. Lazebnik, C. Schmid, and J. Ponce, “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories,” in *CVPR*, 2006, pp. 2169–2178.

[15] J. Liu and M. Shah, “Scene modeling using co-clustering,” in *ICCV*, 2007.

[16] M. Marszałek, C. Schmid, H. Harzallah, and J. van de Weijer, “Learning object representations for visual object class recognition, pascal voc 2007.”

[17] K. Mikolajczyk, B. Leibe, and B. Schiele, “Multiple object class detection with a generative model,” in *CVPR*, 2006.

[18] A. Mojsilović, J. Gomes, and B. Rogowitz, “Semantic-friendly indexing and quering of images based on the extraction of the objective semantic cues,” *Int. J. Comput. Vision*, vol. 56, no. 1-2, 2004.

[19] F. Moosmann, E. Nowak, and F. Jurie, “Randomized clustering forests for image classification,” *TPAMI*, vol. 30, no. 9, 2008.

[20] E. Nowak, F. Jurie, and B. Triggs, “Sampling strategies for bag-of-features image classification,” in *ECCV*, 2006.

[21] F. Perronnin, “Universal and adapted vocabularies for generic visual categorization,” *TPAMI*, vol. 30, no. 7, 2008.

[22] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, “Lost in quantization: Improving particular object retrieval in large scale image databases,” in *CVPR*, 2008.

[23] P. Quelhas, F. Monay, J. Odobez, D. Gatica-Perez, and T. Tuytelaars, “A thousand words in a scene.” *TPAMI*, vol. 29, no. 9, 2007.

[24] K. van de Sande, T. Gevers, and C. Snoek, “Evaluation of color descriptors for object and scene recognition,” in *CVPR*, 2008.

[25] J. Sivic and A. Zisserman, “Video Google: A text retrieval approach to object matching in videos,” in *ICCV*, 2003.

[26] E. Sudderth, A. Torralba, W. Freeman, and A. Willsky, “Describing visual scenes using transformed objects and parts,” *IJCV*, vol. 77, no. 1-3, 2008.

[27] M. Tahir, K. van de Sande, J. Uijlings, F. Yan, X. Li, K. Mikolajczyk, J. Kittler, T. Gevers, and A. Smeulders, “Surreyuva_srkda method, pascal voc 2008,” http://pascallin.ecs.soton.ac.uk/ challenges/VOC/voc2008/workshop/tahir.pdf.

[28] T. Tuytelaars and C. Schmid, “Vector quantizing feature space with a regular lattice,” in *ICCV*, 2007.

[29] J. Vogel and B. Schiele, “Semantic modeling of natural scenes for content-based image retrieval,” *IJCV*, vol. 72, pp. 133–157, 2007.

[30] J. Winn, A. Criminisi, and T. Minka, “Object categorization by learned universal visual dictionary,” in *ICCV*, 2005, pp. 1800–1807.

[31] L. Yang, R. Jin, C. Pantofaru, and R. Sukthankar, “Discriminative cluster refinement: Improving object category recognition given limited training data,” in *CVPR*, 2007.

[32] H. Jégou, M. Douze, and C. Schmid, “On the burstiness of visual elements,” in *CVPR*, 2009.

[33] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, August 2006.

[34] D. Blei, A. Ng, and M. Jordan, “Latent dirichlet allocation,” *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003.

[35] T. Hofmann, “Unsupervised learning by probabilistic latent semantic analysis.” *Machine Learning*, vol. 42, no. 1/2, pp. 177–196, 2001.

[36] B. Silverman and P. Green, *Density Estimation for Statistics and Data Analysis*. London: Chapman and Hall, 1986.

[37] N. Vasconcelos and A. Lippman, “A unifying view of image similarity,” in *ICPR*, 2000, pp. 1038–1041.

[38] D. Lowe, “Distinctive image features from scale-invariant keypoints,” *IJCV*, vol. 60, 2004.

[39] L. Fei-Fei, R. Fergus, and P. Perona, “Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories,” in *WGMBV*, 2004.

[40] G. Griffin, A. Holub, and P. Perona, “Caltech-256 object category dataset,” Caltech, Tech. Rep. UCB/CSD-04-1366, 2007.

[41] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results,” http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[42] ——, “The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results,” http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html.

[43] A. Oliva and A. Torralba, “Modeling the shape of the scene: A holistic representation of the spatial envelope,” *IJCV*, vol. 42, no. 3, 2001.

[44] G. Burghouts, A. Smeulders, and J. Geusebroek, “The distribution family of similarity distances,” in *NIPS*, 2007.

Jan C. van Gemert was a BSc student in Eindhoven at Fontys and a MSc and PhD student at the University of Amsterdam. He interned at MERL in Cambridge, MA, for half a year and spent time as a visiting researcher in the National Institute of Informatics in Tokyo, Japan. Currently he holds a post-doctoral position at École Normale Supérieure in Paris with Jean Ponce. His research interests first responded to low-level visual features after which they were classified as image and video categorization and are now gaining depth with 3D modeling.

Cor J. Veenman received the MSc. degree in computer science from the Free University in Amsterdam and the PhD degree from the Delft University of Technology in Delft. He worked as assistant professor in the Man-Machine Interaction group at the Delft University of Technology. Since 2005 he is appointed assistant professor in computational forensic science at the University of Amsterdam and at the same time he is affiliated with the Netherlands Forensic Institute, The Hague. He researches supervised learning problems for mining and biometrics applications in the digital forensics domain.

Arnold W.M. Smeulders graduated from Technical University of Delft in physics in 1977 (MSc) and in 1982 from Leiden University in medicine (PhD) on the topic of visual pattern analysis. He is scientific director of the Intelligent Systems Lab Amsterdam, of MultimediaN the Dutch public-private partnership, and of the ASCI national research school. He participates in the EU-Vision, DELOS and MUSCLE networks of excellence. He is fellow of the International Association of Pattern Recognition. His research interest is in cognitive vision, content-based image retrieval, learning and tracking, and the picture-language question. He has written 300 papers in refereed journals and conferences and graduated 32 PhD-students. The ISIS research group concentrates on theory, practice and implementation of multimedia information analysis including image databases and computer vision. The group has an extensive record in co-operations with Dutch institutions and industry in the area of multimedia and video analysis. Currently he is an associated editor of the *International Journal of Computer Vision* and the *IEEE Transactions on Multimedia*.

Jan-Mark Geusebroek is an assistant professor in the Intelligent Systems Lab at the University of Amsterdam. His research interest is in the meaningful representation of pictorial content for computer vision. He has authored and co-authored many papers on the topics of color, texture, and natural image statistics. Geusebroek received his PhD in computer science from the University of Amsterdam and was awarded a prestigious young talent grant from the Netherlands Organization for Scientific Research.