Kernel Codebooks for Scene Categorization

Jan C. van Gemert, Jan-Mark Geusebroek,
Cor J. Veenman, and Arnold W.M. Smeulders

Intelligent Systems Lab Amsterdam (ISLA),
University of Amsterdam,
Kruislaan 403, 1098 SJ, Amsterdam,
The Netherlands
J.C.vanGemert@gmail.com
{J.M.Geusebroek,C.J.Veenman,ArnoldSmeulders}@uva.nl

Abstract. This paper introduces a method for scene categorization by modeling ambiguity in the popular codebook approach. The codebook approach describes an image as a bag of discrete visual codewords, where the frequency distributions of these words are used for image categorization. There are two drawbacks to the traditional codebook model: codeword uncertainty and codeword plausibility. Both of these drawbacks stem from the hard assignment of visual features to a single codeword. We show that allowing a degree of ambiguity in assigning codewords improves categorization performance for three state-of-the-art datasets.

1 Introduction

This paper investigates automatic scene categorization, which focuses on the task of assigning images to predefined categories. For example, an image may be categorized as a beach, office or street scene. Applications of automatic scene categorization may be found in content-based retrieval, object recognition, and image understanding.

One particular successful scene categorization method is the codebook approach. The codebook approach is inspired by a word-document representation as used in text retrieval, first applied on images in texture recognition [1]. The codebook approach allows classification by describing an image as a bag of features, where image features, typically SIFT [2], are represented by discrete visual prototypes. These prototypes are defined beforehand in a given vocabulary. A vocabulary is commonly obtained by following one of two approaches: an annotation approach or a data-driven approach. The annotation approach obtains a vocabulary by assigning meaningful labels to image patches [3–5], for example sky, water, or vegetation. In contrast, a data-driven approach applies vector quantization on the features using k-means [6–11] or radius-based clustering [12]. Once a vocabulary is obtained, this vocabulary is employed by the codebook approach to label each feature in an image with its best representing codeword. The frequency of these codewords in an image form a histogram which is subsequently used in a scene categorization task.

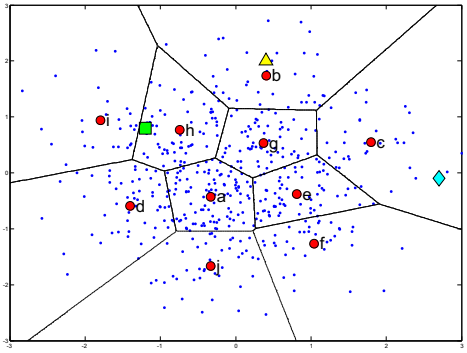

Fig. 1. An example showing the problems of codeword ambiguity in the codebook model. The small dots represent image features, the labeled red circles are codewords found by unsupervised clustering. The triangle represents a data sample that is well suited to the codebook approach. The difficulty with codeword uncertainty is shown by the square, and the problem of codeword plausibility is illustrated by the diamond.

One drawback of the codebook approach is the hard assignment of codewords in the vocabulary to image feature vectors. This may be appropriate for text but not for sensory data with large variety in appearance. The hard assignment gives rise to two issues: *codeword uncertainty* and *codeword plausibility*. Codeword uncertainty refers to the problem of selecting the correct codeword out of two or more relevant candidates. The codebook approach merely selects the best representing codeword, ignoring the relevance of other candidates. The second drawback, codeword plausibility denotes the problem of selecting a codeword without a suitable candidate in the vocabulary. The codebook approach assigns the best fitting codeword, regardless the fact that this codeword is not a proper representative. Figure 1 illustrates both these problems. Accordingly, the hard assignment of codewords to image features overlooks codeword uncertainty, and may label image features by non-representative codewords.

We propose an uncertainty modeling method for the codebook approach. In effect, we apply techniques from kernel density estimation to allow a degree of ambiguity in assigning codewords to image features. We argue that retaining ambiguity between features is a more suitable representation than hard assignment of a codeword to an image feature. By using kernel density estimation, the uncertainty between codewords and image features is lifted beyond the vocabulary and becomes part of the codebook model.

This paper is organized as follows. The next section gives an overview of the related literature on codebook-based scene categorization. Section 3 introduces four types of ambiguity in the codebook model. We show the performance of our method on three datasets in Sect. 4. Finally, Sect. 5 concludes the paper.

2 Related Work

The traditional codebook approach [1, 13] treats an image as a collection of local features where each feature is represented by a codeword from the codebook vocabulary. One extension of the traditional codebook approach aims to capture co-occurrences between codewords in the image collection. Typically, this co-occurrence is captured with a generative probabilistic model [14, 15]. To this end, Fei-Fei and Perona [7] introduce a Bayesian hierarchical model for scene categorization. Their goal is a generative model that best represents the distribution of codewords in each scene category. They improve on Latent Dirichlet Allocation (LDA) [15] by introducing a category variable for classification. The proposed algorithm is tested on a dataset of 13 natural scene categories where it outperforms the traditional codebook approach by nearly 30%. The work by Fei-Fei and Perona is extended by Quelhas *et al.* [11], who investigate the influence of training data size. Moreover, Bosch *et al.* [6] show that probabilistic latent semantic analysis (pLSA) improves on LDA. The contributions on codeword ambiguity in this paper are easily extended with co-occurrence modeling.

Besides co-occurrence modeling, other improvements on the codebook approach focus on the vocabulary. A semantic vocabulary inspired by Oliva and Torralba [16] is presented by Vogel and Schiele [5]. The authors construct a vocabulary by labeling image patches with a semantic label, for example sky, water or vegetation. The effectiveness of this semantic codebook vocabulary is shown in a scene categorization task. Moreover, a similar approach [4] provides the basis for the successful results on TRECVID news video by Snoek *et al.* [17], who draw inspiration from Naphade and Huang [18]. Furthermore, Winn *et al.* [19] concentrate on a universal codebook vocabulary, whereas Perronnin *et al.* [10] focus on class-specific vocabularies. In contrast to annotating a vocabulary, Jurie and Triggs [12] compare clustering techniques to obtain a data-driven vocabulary. Specifically, they show that radius-based clustering outperforms the popular k-means clustering algorithm, and we will make use of this observation below.

Since the codebook approach treats an image as a histogram of visual words, the spatial structure between words is lost. Spatial structure is incorporated by Lazebnik *et al.* [8] who extend the work of Grauman and Darrell [20] with a spatial pyramid matching scheme. Furthermore research on incorporating spatial information in the codebook model focuses on regions of interest [21], object segmentation [22], and shape masks [23]. To demonstrate the modularity of our work, we incorporate spatial pyramid matching because of the excellent performance reported by Lazebnik *et al.* [8].

3 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

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

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. Basically, an image is represented 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 [24]. 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

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} K_\sigma\left(x - X_i\right), \tag{2}$$

where n is the total number of samples and X_i is the value of sample i.

Kernel density estimation requires 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 [14]. In this paper we use the SIFT descriptor that draws on the Euclidian distance as its distance function [2]. The Euclidian 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

$$K_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{1}{2}\frac{x^2}{\sigma^2})\ . \tag{3}$$

The Gaussian kernel assumes that the variation between a data sample and a codeword may be described by a normal distribution. This normal distribution requires a scale parameter σ which determines the size of the kernel. The kernel size needs to be tuned to the appropriate degree of smoothing between data samples. This smoothing determines the degree of similarity between data samples, and is dependent on the dataset, the feature length, and the range of the feature values. These dependencies change for various datasets. Therefore, in the experiments we will tune the kernel size by cross-validation. In summary, the size of the kernel depends on the data 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 (like the Gaussian kernel) allows kernel density estimation to become part of the codewords, instead of the data samples. Specifically, when the used kernel is symmetric, $K_\sigma(x - X_i) = K_\sigma(X_i - x)$, it trivially follows that there is no effective 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

Table 1. The relationship between various forms of codeword ambiguity and their properties.

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

if the centre of the kernel coincides with the data sample. Hence, a symmetric kernel allows transferring the kernel from the data samples to the codewords, yielding a *kernel codebook*,

$$\mathrm{KCB}(w) = \frac{1}{n}\sum_{i=1}^{n} K_\sigma\left(D(w, r_i)\right), \tag{4}$$

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.

In essence, a kernel codebook smoothes the hard mapping of features in an image region to the codeword vocabulary. This smoothing 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*,

$$\mathrm{UNC}(w) = \frac{1}{n}\sum_{i=1}^{n} \frac{K_\sigma\left(D\left(w, r_i\right)\right)}{\sum_{j=1}^{|V|} K_\sigma\left(D(v_j, r_i)\right)}, \tag{5}$$

distributes a constant amount of probability mass to all relevant codewords, where 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*,

$$\mathrm{PLA}(w) = \frac{1}{n}\sum_{i=1}^{n} \begin{cases} K_\sigma\left(D(w, r_i)\right) & \text{if } w = \arg\min_{v \in V}(D(v, r_i)); \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

selects for an image region r_i the best fitting codeword w and gives that word an amount of mass corresponding to the kernel value of that codeword. Hence, codeword plausibility will give a higher weight to more relevant data samples, however cannot select multiple codeword candidates. The relation between codeword plausibility, codeword uncertainty, the kernel codebook model, and the traditional codebook model is indicated in Table 1.

An example of the weight distributions of the types of codeword ambiguity with a Gaussian kernel is shown in Fig. 2(a). Furthermore, in Fig. 2(b) we

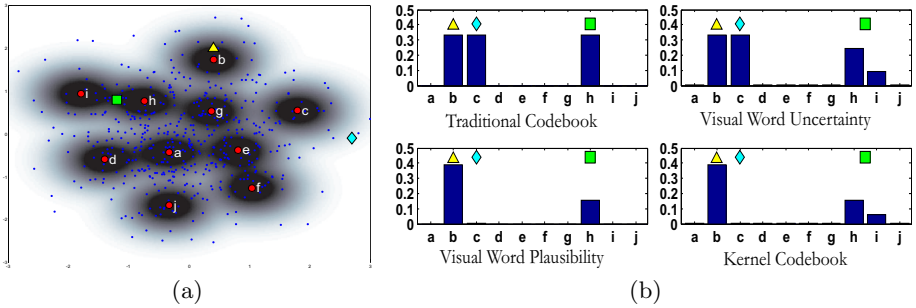

Fig. 2. (a) An example of the weight distribution of a kernel codebook with a Gaussian kernel, where the data and the codewords are taken from Fig. 1. (b) Various codeword distributions, according to Table 1, corresponding to different types of codeword ambiguity. These distributions are based on the kernels shown in Fig. 2(a), where the square, diamond and triangle represent the image features.

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 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 Sect. 4.

The ambiguity between codewords will likely be influenced by the number of words in the vocabulary. When the vocabulary is small, essentially different image parts will be represented by the same vocabulary element. On the other hand, a large vocabulary allows more expressive power, which will likely benefit the hard assignment of the traditional codebook. Therefore, we speculate that codeword ambiguity will benefit smaller vocabularies more than larger vocabularies. We will experimentally investigate the vocabulary size in Sect 4.

Since codewords are image descriptors in a high-dimensional feature space, we envision a relation between codeword ambiguity and feature dimensionality. With a high-dimensional image descriptor, codeword ambiguity will probably 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, most feature points will be close to the boundary between codewords and thus introduces ambiguity between codewords. See Bishop's textbook on pattern recognition and machine learning [14, Chapter 1, pages 33–38] for a thorough explanation and illustra-

tion of the curse of dimensionality. Consequently, increasing the dimensionality of the image descriptor may increase the level of codeword ambiguity. Therefore, our improvement over the traditional codebook model should become more pronounced in a high-dimensional feature space. We will experimentally investigate the effects of the dimensionality of the image descriptor in the next section.

4 Experiments

We experimentally compare codeword ambiguity modeling against the traditional codebook approach for three large and varied datasets: fifteen natural scene categories from Lazebnik *et al.* [8], Caltech-101 by Fei-Fei and Perona [25], Caltech-256 by Griffin *et al.* [26]. 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. For our experimental setup we closely follow Lazebnik *et al.* [8]. We follow this work since it has shown excellent performance on these datasets.

4.1 Experimental Setup

To obtain reliable results, we repeat the experimental process 10 times. Thus, 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. This 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 a SVM with a histogram intersection kernel. Specifically, we use libSVM [27], 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 of the codebook kernel. The classification rate we report is the average of the per-class recognition rates which in turn are averaged over the 10 random test sets.

For image features we again follow Lazebnik *et al.* [8], and use a SIFT descriptors sampled on a regular grid. A grid has been shown to outperform interest point detectors in image classification [7, 12, 9]. Hence, we compute all SIFT descriptors on 16x16 pixel patches, computed over a dense grid sampled every 8 pixels.

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

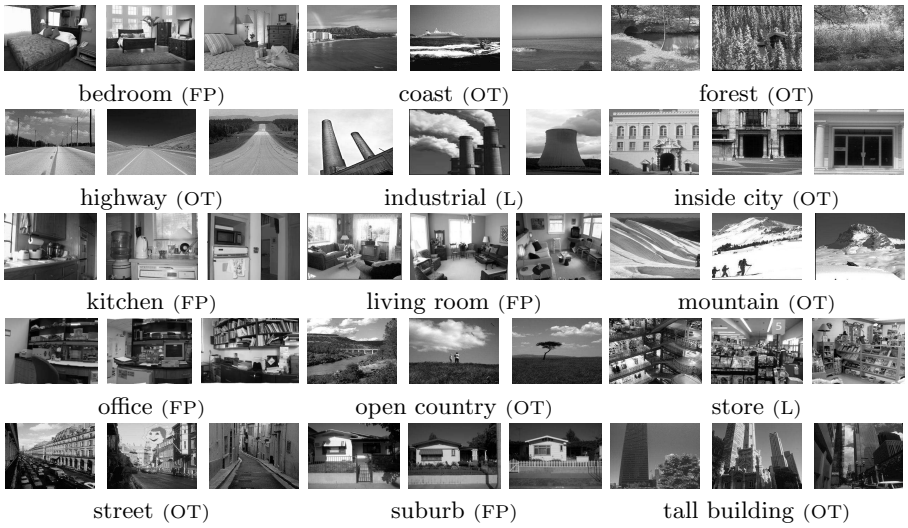

Fig. 3. Example images from the Scene-15 dataset. Each category is labeled with the annotator, where (OT) denotes Oliva and Torralba [16], (FP) is Fei-Fei and Perona [7], and (L) refers to Lazebnik *et al.* [8].

4.2 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 [7, 8, 16]. The Scene-15 dataset consists of 4485 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 Fig. 3 we show examples of the scene dataset. We use an identical experimental setup as Lazebnik *et al.* [8], and select 100 random images per category as a train set and the remaining images as the test set.

We start the experiments with an in-depth analysis of the types of codeword ambiguity, vocabulary size and feature dimensionality. To evaluate feature dimensionality 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 and 60 dimensions. However, because of space constraints we omit the results for the 60-dimensional features since they show the same trend as the other dimensions. 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 Fig. 4.

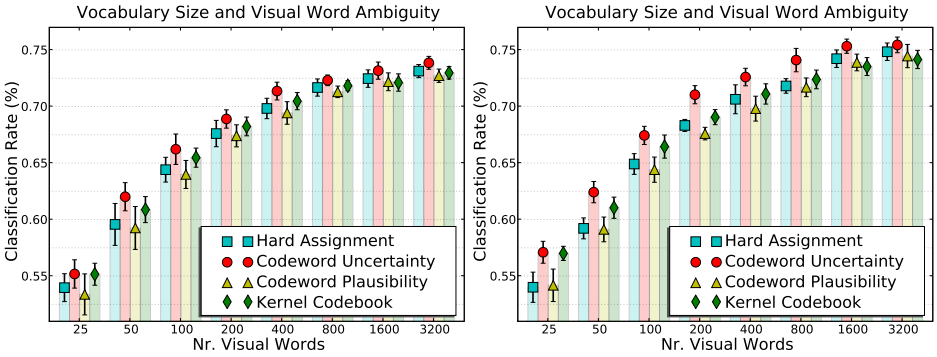

Fig. 4. Classification performance of various types of codeword ambiguity for the Scene-15 dataset over various vocabulary sizes and feature dimensions.

We start the analysis of the results in Fig. 4 with the various types of codeword ambiguity. The results show that codeword uncertainty outperforms all 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 it can be seen that codeword uncertainty already outperforms hard assignment with a 400-word vocabulary, and this trend holds for larger vocabulary size pairs. On the other end of the performance scale there is codeword plausibility, which always yields the worst results. A kernel codebook outperforms hard assignment for smaller vocabulary sizes, however for larger vocabularies hard assignment performs equally well. These differences between codeword ambiguity types become more pronounced when using a smaller vocabulary, whereas using a larger vocabulary evens out the results between ambiguity types. Additionally, 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 asymptotically improves performance.

To gain insight in the performance variation between the various types of codeword ambiguity we show the overlap percentage between the predicted class labels for all paired method in Fig. 5. The first thing that is striking in Fig. 5, is the high class 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, 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. On the other end of the spectrum we find the kernel

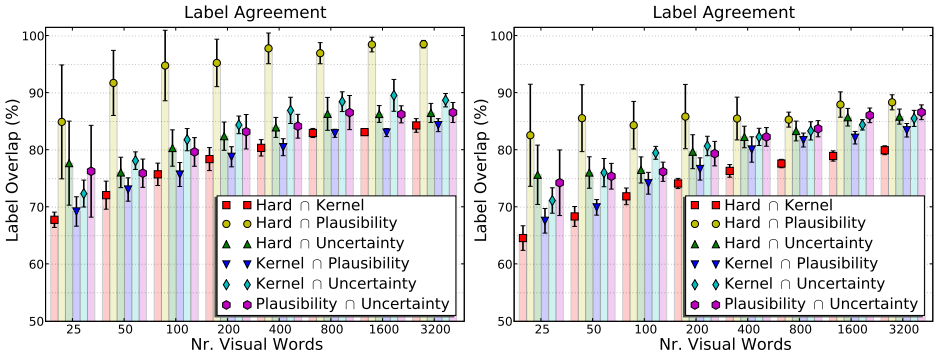

Fig. 5. Comparing the overlap of the class labels as predicted by various types of codeword ambiguity for the Scene-15 dataset.

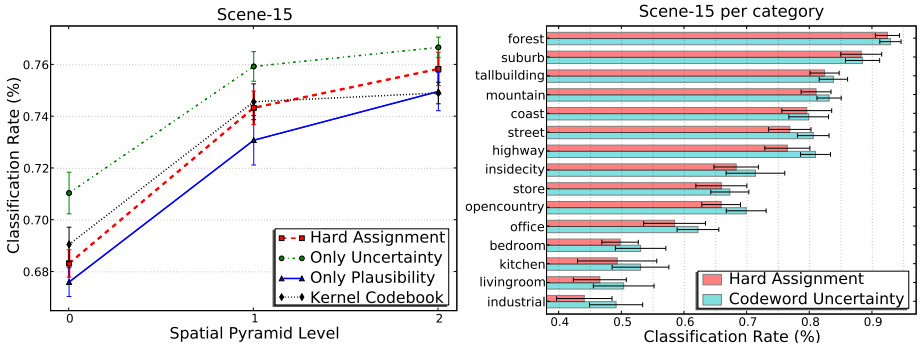

Fig. 6. Comparing the performance on the Scene-15 dataset of various types of codeword ambiguity using the spatial pyramid (left), and per category (right).

codebook and hard assignment pair, which share the least number of class labels. This low label overlap may be expected, since these two types represent the extremes of the types of codeword ambiguity. 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 class labels. This reduced label overlap in a high-dimensional space may be explained by the increased effectiveness of codeword ambiguity in a high-dimensional space. A further trend in label overlap is 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 for all methods to those images that can be predicted correctly by using a larger vocabulary. This link between increased performance and increased class label overlap also explains that the class label overlap is generally high between all types of codeword ambiguity.

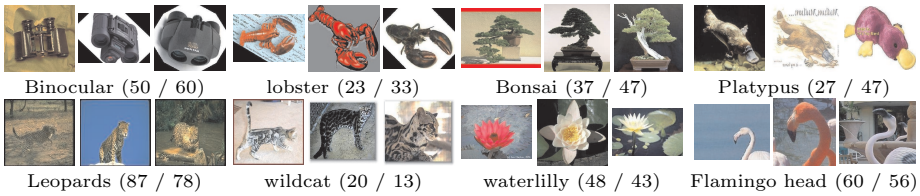

Fig. 7. Examples of the Caltech-101 set. Top: the top 4 classes where our method improves most, Bottom: the 4 classes where our method decreases performance. The numbers in brackets indicate the classification rate (hard / uncertainty).

Fig. 8. Examples of the Caltech-256 set. Top: the top 4 classes where our method improves most, Bottom: the 4 classes where our method decreases performance most. The numbers in brackets indicate the classification rate (hard / uncertainty).

To show the modularity of our approach and improve results we incorporate the spatial pyramid by Lazebnik *et al.* [8]. The spatial pyramid divides an image into a multi-level pyramid of increasingly fine subregions and computes a codebook descriptor for each subregion. We use the 128 dimensional feature size since this gives the best results. Moreover, we find a vocabulary of 200 codewords, since this number is also used by Lazebnik *et al.* [8]. The results for the various forms of codeword ambiguity for the first two levels of the spatial pyramid are shown in Fig. 6. Note that the codeword uncertainty outperforms the hard assignment of the traditional codebook for all levels in the pyramid. Moreover, codeword uncertainty at pyramid level 1 already outperforms the traditional codebook at pyramid level 2. 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. The classification results for level 0 of the pyramid, split out per category are shown in Fig. 6. Note that by using codeword uncertainty the performance of all categories are similar or improve upon a traditional codebook.

Due to small implementation differences, our re-implementation of the original paper [8] 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.

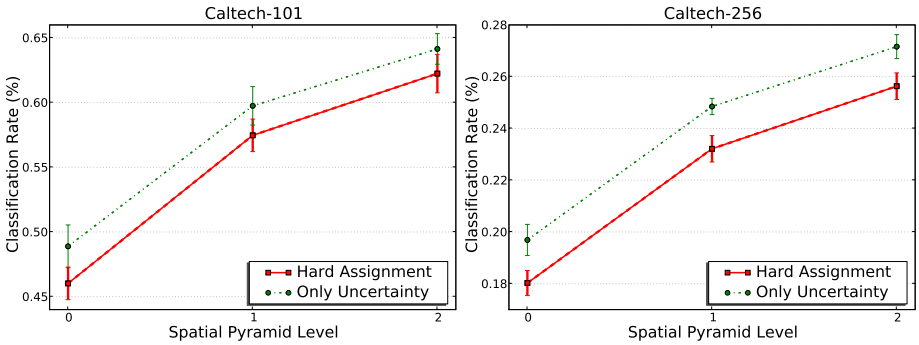

Fig. 9. Classification performance of Caltech-101 (left) and Caltech-256 (right).

4.3 Experiment 2 and 3: Caltech-101 and Caltech-256

Our second set of experiments are done on the Caltech-101 [25] and Caltech-256 [26] datasets. The Caltech-101 dataset contains 8677 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 Fig. 7 we show some example image of the Caltech-101 set. Some of the problems of Caltech-101 are solved by the Caltech-256 dataset. The Caltech-256 dataset holds 29780 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 to face one direction. We report classification performance on both sets.

Our experimental results for both the Caltech-101 as Caltech-256 are generated by using 30 images per category for training. For testing, we used 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 [26, 8]. We use 128 dimensions, and compare the traditional hard assignment with codeword uncertainty since this has shown to give the best results on the Scene-15 dataset. The classification results per spatial pyramid level are shown in Fig. 9. For both sets, the codeword uncertainty method outperforms the traditional codebook.

4.4 Summary of Experimental Results

The experiments on the Scene-15 dataset in figures 4 and 6 show that codeword plausibility hurts performance. Codeword plausibility is dominated by those few image features that are closest to a codeword. In essence, codeword plausibility ignores 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. Therefore, codeword uncertainty yields the best results, since it models ambiguity between codewords, without taking codeword plausibility into account.

The results in Fig. 4 indicate that codeword ambiguity is more effective for higher dimensional features than for lower dimensions. We attribute this to an increased robustness to the curse of dimensionality. The curse prophesizes that increasing the dimensionality will increase the fraction of feature vectors on or near the boundary of codewords. Hence, increasing the dimensionality will increase codeword uncertainty. Furthermore, Fig. 4, shows that a larger vocabulary mostly benefits hard assignment, and asymptotically increases performance. Thus, since our ambiguity modeling approach starts with a higher performance, it stands to reason that our model will reach the maximum performance sooner.

Table 2. The relationship between the data set size and the relative performance of codeword uncertainty over hard assignment for 200 codewords.

Data set	**Train set size**	**Test set size**	**Performance Increase (%)**
Scene-15	1500	2985	4.0 ± 1.7 %
Caltech-101	3030	5050	6.3 ± 1.9 %
Caltech-256	7680	6400	9.3 ± 3.0 %

The results over the Scene-15, Caltech-101, and Caltech-256 datasets are summarized in Table 2. This table shows the relative improvement of codeword uncertainty over hard assignment. 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. What is more, the results of all experiments show that codeword uncertainty outperforms the traditional hard assignment over all dimensions, all vocabulary sizes, and over all datasets.

5 Conclusion

This paper presented a fundamental improvement on the popular codebook model for scene categorization. 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 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. We have demonstrated the viability of our approach by improving results on recent codebook methods. These results are shown on three state-of-the-art datasets, where our method consistently improves over the traditional codebook model. What is more, we found that our ambiguity modeling approach suffers less from the curse of dimensionality, reaping higher benefits in a high-dimensional feature space. Furthermore, with an increasing number of scene categories, the effectiveness of our method becomes

more pronounced. Therefore, as future image features and datasets are likely to increase in size, our ambiguity modeling method will have more and more impact.

References

1. Leung, T., Malik, J.: Representing and recognizing the visual appearance of materials using three-dimensional textons. IJCV **43** (2001) 29–44
2. Lowe, D.: Distinctive image features from scale-invariant keypoints. IJCV **60** (2004) 91–110
3. Boutell, M., Luo, J., Brown, C.: Factor-graphs for region-based whole-scene classification. In: CVPR-SLAM. (2006)
4. van Gemert, J., Geusebroek, J., Veenman, C., Snoek, C., Smeulders, A.: Robust scene categorization by learning image statistics in context. In: CVPR-SLAM. (2006)
5. Vogel, J., Schiele, B.: Semantic modeling of natural scenes for content-based image retrieval. IJCV **72** (2007) 133–157
6. Bosch, A., Zisserman, A., Munoz, X.: Scene classification using a hybrid generative/discriminative approach. TPAMI **30** (2008) 712–727
7. Fei-Fei, L., Perona, P.: A bayesian hierarchical model for learning natural scene categories. In: CVPR. (2005)
8. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR. (2006) 2169–2178
9. Nowak, E., Jurie, F., Triggs, B.: Sampling strategies for bag-of-features image classification. In: ECCV. (2006)
10. Perronnin, F., Dance, C., Csurka, G., Bressan, M.: Adapted vocabularies for generic visual categorization. In: ECCV. (2006)
11. Quelhas, P., Monay, F., Odobez, J., Gatica-Perez, D., Tuytelaars, T., Gool, L.V.: Modeling scenes with local descriptors and latent aspects. In: ICCV. (2005)
12. Jurie, F., Triggs, B.: Creating efficient codebooks for visual recognition. In: ICCV. (2005) 604–610
13. Sivic, J., Zisserman, A.: Video Google: A text retrieval approach to object matching in videos. In: ICCV. Volume 2. (2003) 1470–1477
14. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2006)
15. Blei, D., Ng, A., Jordan, M.: Latent dirichlet allocation. J. Mach. Learn. Res. **3** (2003) 993–1022
16. Oliva, A., Torralba, A.: Modeling the shape of the scene: A holistic representation of the spatial envelope. IJCV **42** (2001) 145–175
17. Snoek, C., Worring, M., van Gemert, J., Geusebroek, J., Smeulders, A.: The challenge problem for automated detection of 101 semantic concepts in multimedia. In: ACM Multimedia. (2006)
18. Naphade, M., Huang, T.: A probabilistic framework for semantic video indexing, filtering, and retrieval. Transactions on Multimedia **3** (2001) 141–151
19. Winn, J., Criminisi, A., Minka, T.: Object categorization by learned universal visual dictionary. In: ICCV. (2005) 1800–1807
20. Grauman, K., Darrell, T.: The pyramid match kernel: discriminative classification with sets of image features. In: ICCV. (2005) 1458–1465
21. Bosch, A., Zisserman, A., Munoz, X.: Image classification using random forests and ferns. In: ICCV. (2007)

22. Larlus, D., Jurie, F.: Category level object segmentation. In: International Conference on Computer Vision Theory and Applications. (2007)
23. Marszałek, M., Schmid, C.: Accurate object localization with shape masks. In: CVPR. (2007)
24. Silverman, B., Green, P.: Density Estimation for Statistics and Data Analysis. Chapman and Hall, London (1986)
25. Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In: WGMBV. (2004)
26. Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset. Technical Report UCB/CSD-04-1366, California Institute of Technology (2007)
27. Chang, C., Lin, C.: LIBSVM: a library for support vector machines. (2001)