LETTER

Scene Categorization with Classified Codebook Model

Xu YANG[†a)], **De XU**[†], **Songhe FENG**[†], **Yingjun TANG**[††], ***Nonmembers***, ***and*** **Shuoyan LIU**[†], ***Student Member***

SUMMARY This paper presents an efficient yet powerful codebook model, named classified codebook model, to categorize natural scene category. The current codebook model typically resorts to large codebook to obtain higher performance for scene categorization, which severely limits the practical applicability of the model. Our model formulates the codebook model with the theory of vector quantization, and thus uses the famous technique of classified vector quantization for scene-category modeling. The significant feature in our model is that it is beneficial for scene categorization, especially at small codebook size, while saving much computation complexity for quantization. We evaluate the proposed model on a well-known challenging scene dataset: 15 Natural Scenes. The experiments have demonstrated that our model can decrease the computation time for codebook generation. What is more, our model can get better performance for scene categorization, and the gain of performance becomes more pronounced at small codebook size.

key words: *codebook model, codebook generation, visual words, classified vector quantization, scene categorization*

1. Introduction

Today, digital images are ubiquitous, which implies the necessity of automatic image indexing by their semantic content. In this paper we address the problem of categorizing an image into one scene category (e.g. forest, coast, street, etc.), that has high potential for many applications, such as image database browsing, image retrieval, and object recognition. In the last decade, the popular *codebook model* (or *bag-of-words model*) [1] has shown excellent performance for visual categorization. Therefore, we focus on modeling scene categories with the codebook model.

The codebook model treats an image as a histogram of the "visual words". Current codebook model has the following typical steps. It begins with extracting local image features (e.g. SIFT [2] or HOG [3]). Then, a visual codebook is generated, which divides the feature space into several regions. After that, local features are quantized into discrete visual words according to the codebook. An image is finally described as the histogram of the visual words, and scene types are categorized by classifiers such as SVM. As far as we know, all previous literatures on the codebook model are consistent with this framework. We refer to the framework as the basic codebook model (BCM).

However, one formidable problem that baffles BCM is the confliction between the codebook size and the discriminative power for scene categories. A too small codebook does not discriminate well between scene categories. Hence, current methods often take several thousands of words to improve the discriminative power of a codebook model. However, large codebook size will severely limit the practical applicability of the model, because of its storage requirement, working memory usage, and the computation time to generate the codebook [4]. We present *classified codebook model* (CCM) for the task of scene categorization, inspired by the theory of classified vector quantization (CVQ) [5]. As illustrated by Fig. 1, local image features have to be classified into several classes before they are quantized into visual words. As will be seen, compared with BCM, CCM needs much less computational time for codebook generation. Furthermore, CCM outperforms BCM for scene categorization especially at small codebook size. All these results are thoroughly verified on well-know challenging scene dataset: 15 natural scenes [6]. We summarize our contributions as follows.

1) We formulate the codebook model with the theory of vector quantization (VQ). Therefore, many famous techniques developed in VQ can be effectively applied to the codebook model.

2) CCM naturally follows the strategy of divide-and-conquer, which greatly reduces the time complexity of codebook generation.

3) We classify image features by their perceptual importance. By allocating more visual words to the features with higher perceptual importance, CCM produces more compact and discriminatively powerful codebook for scene

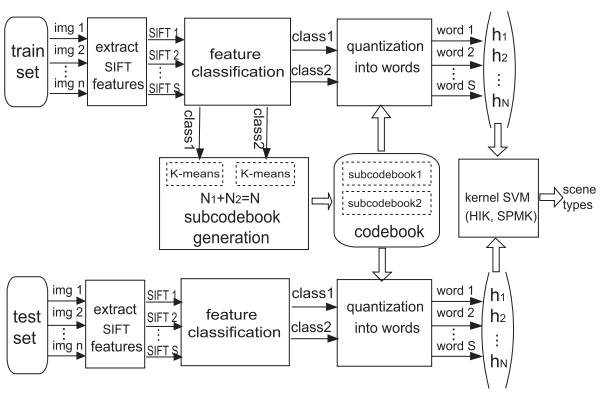

Fig. 1 Framework of CCM.

Manuscript received January 7, 2011.

[†]The authors are with the School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China.

[††]The author is with the Software School, Jiangxi University of Finance and Economics, Jiangxi China.

a) E-mail: yangxubj@126.com

DOI: 10.1587/transinf.E94.D.1349

category.

The paper is organized as follows. In Sect. 2, we formulate the codebook model in context with vector quantization, which is followed by the problem examination of BCM. Section 3 describes the principle of the proposed model, and the key steps involved in the proposed model. Section 4 presents the experimental results. Finally, this paper is concluded in Sect. 5.

2. Formulation of the Codebook Model

2.1 Vector Quantization Formulation

We formulate the codebook model from the perspective of vector quantization [7]. VQ is a coding technique that maps a feature vector of k-dimensional space $\boldsymbol{R}^k$ to a finite set of vectors $\mathcal{V}$: $\boldsymbol{R}^k \rightarrow \mathcal{V}$, where $\mathcal{V} = \{\boldsymbol{v}_1, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_N\}, \boldsymbol{v}_i \in \boldsymbol{R}^k, i \in \{1, 2, \ldots, N\}$. In the codebook model, each $\boldsymbol{v}_i$ is defined as a visual word, and the set $\mathcal{V}$ is the visual codebook, N is the size of the codebook. Particularly, for the model of BCM, vector quantization maps each local feature $\boldsymbol{x}$ to one visual word by nearest neighbor matching, namely,

$$VQ(\boldsymbol{x}, \mathcal{V}) = \boldsymbol{v}_j \Leftrightarrow dist(\boldsymbol{x}, \boldsymbol{v}_j) < dist(\boldsymbol{x}, \boldsymbol{v}_i) \tag{1}$$

where $i \in \{1, 2, \ldots, N\}, i \neq j$. The final representation of an image X on codebook $\mathcal{V}$ is the frequency histogram of visual words.

2.2 Problem Examination of BCM

For BCM, current methods typically use clustering algorithms to generate codebook. Particularly, K-means clustering often serves as default algorithm due to mainly its simplicity. As noted by Jurie [8] and Gemert [9], K-means tends to place clusters near the most frequently occurring features. However, features that occurs frequently are not necessarily the most discriminative for scene categories. For example, smooth regions (e.g. skies, walls, roads) occur far more frequently than others region with detail perceptual features (e.g. edges, faces, signs), but over-frequent features located in the smooth regions typically contain much less information than the middle or low frequency features. In other words, over-frequent features should be allocated less visual words compared with other informative features.

Another problem suffered by BCM is the time complexity of codebook generation. For large database, the visual codebook usually contains tens of thousands of visual words, and the scale of training features will reach tens of millions. Under the framework of BCM, the time complexity is a nightmare.

3. Classified Codebook Model (CCM)

3.1 Principle of CCM

We present one novel framework of codebook model by using the theory classified vector quantization (CVQ) [5]. In the field of vector quantization, CVQ has shown excellent performance for image compression at low bit rates. The theory of CVQ is simple: A vector quantizer benefits from classifying information source into several subsources [5]. In this paper we extend its application to the codebook model, and call the poposed model as classified codebook model. Figure 1 illustrates the main process of CCM. The key steps involved in CCM are as follows.

3.2 Feature Classification

To clearly demonstrate the approach, we simply classify input features into two classes: smooth class and edge class. We take dense SIFT descriptor [2] as local image feature, that describes the information of orientation distribution of gradients for one local patch of an image. Therefore, to classify one descriptor, we just need to classify its corresponding patch, by measuring the variance of gradient of that patch. For each SIFT descriptor, we create a $w \times w$ window as its working patch, and define the variance of gradient magnitude of the patch as follow:

$$\sigma = \frac{1}{w \times w} \sqrt{\sum_{x=1}^{w} \sum_{y=1}^{w} (m(x, y) - \overline{m})^2} \tag{2}$$

where $m(x, y)$ is the gradient magnitude for pixel at location (x, y), and $\overline{m}$ is the average of magnitude in the patch. The classification rule for a descriptor is defined as

$$s(\sigma) = \begin{cases} \text{smooth class} & \text{if } \sigma \leq th\,, \\ \text{edge class} & \text{otherwise} \end{cases} \tag{3}$$

where th is a threshold determined empirically by user. Our experiments indicate that lower th value results in more false detection for edge class, while higher th value leads to miss detection for edge class. However, for simplicity we always set $th = 0.001$ in our experiment.

3.3 Subcodebook Generation and Quantizaiton

As stated in Sect. 2, BCM tends to assign too many visual words to smooth class. However, CCM offers a direct way to allocate small subcodebook size to the smooth class, but large size to the edge class. By employing K-means clustering to each class, we generate two subcodebooks, namely, $\mathcal{V}_1$(for smooth class) and $\mathcal{V}_2$(for edge class), with the size of N_1 and N_2 respectively. In experiments we always set the parameters N_1 and N_2 so as to $\frac{N_2}{N_1+N_2} = 0.8$. In CCM, the quantization step performs in the same manner as BCM (see Eq. (1)), except that local features belonging to one class are quantized only into visual words belonging to the same class. For a input feature $\boldsymbol{x}$, feature classification is employed to determine which class the feature belongs to. If $\boldsymbol{x}$ belongs to class i, then the ith subcodebook is employed to quantize $\boldsymbol{x}$ by nearest neighbor matching:

$$CVQ(\boldsymbol{x}, \mathcal{V}) = \begin{cases} VQ(\boldsymbol{x}, \mathcal{V}_1) & \text{if } \boldsymbol{x} \in \text{smooth class} \\ VQ(\boldsymbol{x}, \mathcal{V}_2) & \text{if } \boldsymbol{x} \in \text{edge class} \end{cases} \tag{4}$$

By contrasting Eq. (1) and Eq. (4), it is clear that CCM reduces the computational complexity of BCM, because it searches only small subcodebooks instead of the entire codebook, which follows the strategy of divide-and-conquer.

4. Experiments

We experimentally compare CCM against BCM on well-know challenging scene dataset: fifteen natural scene categories from Lazebnik et al. [6]. The dataset consists of 4485 images spread over 15 categories, ranging from natural scenes like mountains and forests to manmade environments like offices and stores. Each category contains 212 $\sim$ 410 images, and the average image size is about 250×300. All the experiments on fifteen scene categories are repeated 5 times. Reported values for all experiments correspond to the average result.

4.1 Local Feature Extraction

This section briefly describes the implementation of extracting features for codebook generation. Following Lazebnik et al. [6], we randomly select 600 images from the database, and compute dense SIFT descriptors on overlapping 16×16 pixel patches (i.e., $w = 16$ in Eq. (2)), over a dense grid sampled over 8 pixels. Consequently, we collect about 1000 descriptors for each image, and totally about 600,000 candidate descriptors. To make the computation practically feasible, we only randomly select 22500 descriptors as training features. Note that BCM and CCM share the same features for codebook generation.

4.2 Codebook Comparison

K-means clustering algorithm is applied to the training features for codebook generation. We first qualitatively compare the codebook generated by BCM and CCM, by visualizing some of local image patches that are clustered together by K-means in Fig. 2. For example, the top-left picture in Fig. 2 has 36 patches, where all of them correspond to *one* visual word in the codebook of BCM. By observing the four pictures on the top, we can see that the patches from smooth class and edge class tends to be mixed together. In contrast, the bottom pictures show that the patches have a more similar appearance, because patches from smooth class and edge class have been well classified before generating the codebook in CCM.

To evaluate the time complexity of codebook generation for BCM and CCM, we compare their computation time under different codebook sizes. The sizes we consider are:{32, 64, 128, 256, 512, 1024}. The experiment is executed by PentiumIV 2.66 GHz machine with 2 GB memory. Note that for CCM we have added the computation cost caused by feature classification, although it spends only small fraction of total computation time. Table 1 shows the results for BCM and CCM at various codebook sizes. As can be seen, the complexity of BCM grows significantly with the codebook size, while CCM retains at a reasonable complexity.

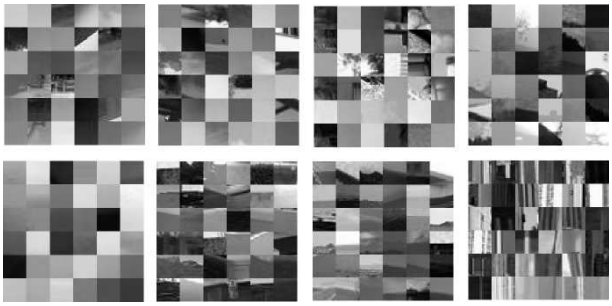

Fig. 2 Codebook comparison. Top: some patches that correspond to visual words from the codebook of BCM. Bottom: ditto, but from the codebook of CCM.

Table 1 Computational time in seconds for codebook generation of BCM and CCM with different codebook sizes.

codebook size	32	64	128	256	512	1024
BCM (sec)	66	288	366	483	597	863
CCM (sec)	60	247	290	361	421	561

4.3 Comparison of Scene-Categorization Performance

We employ support vector machine (SVM) classifier with either histogram intersection kernel (HIK) or spatial pyramid matching kernel (SPMK) for scene categorization, because SVM with HIK and SPMK has shown excellent performance [6]. The complexity for HIK and SPMK computation is $O(n^3 \times N)$, where n is number of images in train set, N is the codebook size. Therefore, to study the practical applicability of CCM, we investigate how the two factors influence the performance of CCM and BCM.

Firstly, to compare the categorization performance for BCM and CCM over fifteen categories for various codebook sizes, we fix $n = 1500$ by randomly selecting 100 images per category to composite the train set, and consider following codebook size: N ={32, 64, 128, 256, 512, 1024}. Table 2 shows the precision for both HIK and SPMK kernel over various codebook sizes. It is shown that CCM consistently outperforms BCM for both HIK and SPMK. For HIK kernel, CCM can easily obtain almost the same performance as BCM with half codebook size. Moreover, it shows a clear advantage of using SMPK over HIK for both BCM and CCM, which is in line with [6] because SPMK essentially captures global spatial information at different resolution.

In Fig. 3 we plot and reproduce the same information as the Table 2. As illustrated in Fig. 3, increasing the codebook size increases the categorization performance for both BCM and CCM, which demonstrate that large codebooks have more discriminative power for scene categorization. However, one significant observation from Fig. 3 is that, although CCM always improves categorization performance, the performance gain of CCM becomes more pronounced

Table 2 Comparison of categorization precision for BCM and CCM over various codebook sizes for two SVM kernels.

codebook size	HIK		SPMK	
	BCM	CCM	BCM	CCM
32	64.50	68.20	75.61	77.29
64	68.50	71.86	78.29	80.30
128	72.09	74.37	80.00	80.57
256	73.83	76.24	80.44	81.07
512	75.71	77.25	80.97	81.44
1024	76.11	78.63	80.77	81.24

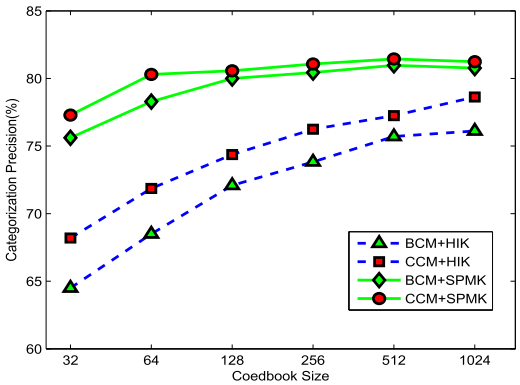

Fig. 3 Categorization performance comparison between BCM and CCM over various codebook sizes for two SVM kernels.

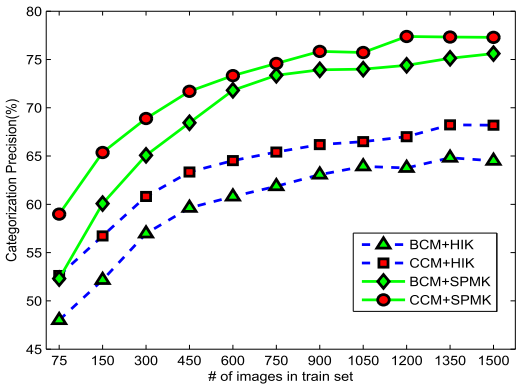

Fig. 4 Comparison between BCM and CCM at codebook size $N = 32$ over various number of images in train set.

for smaller codebook size. For example, when using SPMK at codebook size 64, CCM achieves largest gain of precision relative to BCM, whose performance surprisingly almost reaches the best performance of BCM with SPMK at codebook size 512. This is due to the fact that for BCM, when codebook size is small, BCM concentrates too many visual words on smooth class, and leads to poor codebook. By contrast, CCM works quite well and generates more discriminative codebook particularly at small codebook size.

As final experiment, to investigate the robustness of CCM at small codebook size, we conduct scene categorization with different amount of images in train set. We set the codebook size $N = 32$, and use eleven train sets with following number of images: n ={75,150,300,450, 600, 750, 900, 1050, 1200, 1350, 1500}. Figure 4 illustrates the results over various train sets. From the figure, we have two observations: 1) CCM consistently outperforms BCM for all train sets and for both SVM kernels. 2) When using smaller train set, both CCM and BCM degrade their performances, but the performance gain of CCM relative to BCM remains fairly stable in this case. When taking all observations into account, the results indicate that CCM can robustly provide the highest possible accuracy using relatively small amount of images in train set and small codebook size.

5. Conclusions

We propose the novel *classified codebook model* (CCM) for scene categorization by using classified vector quantization. The proposed model separates local features into two classes (smooth and edge) and generates two subcodebooks to form the whole codebook. Compared to BCM, CCM naturally follows the divide-and-conquer strategy and thus reduces the time complexity of codebook generation. What is more, by allocating more visual words to the informative edge class, CCM can greatly increase the discriminative power for scene categories especially at small codebook size. The experiments show that CCM outperforms BCM in both time complexity and scene-categorization performance. However, in this paper only two classes are considered because we barely focus on validating our model. In the future we will introduce more complicated schemes of feature classification in CCM, which should certainly improve the performance of CCM further.

Acknowledgement

This work is supported by National Nature Science Foundation of China (60803072 and 90820013).

References

[1] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," Proc. ICCV, pp.1470–1477, 2003.
[2] D.G. Lowe, "Distinctive image features from scale-invariant keypoints," Int. J. Comput. Vis., vol.60, no.2, pp.91–110, 2004.
[3] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," Proc. CVPR, vol.1, pp.886–893, 2005.
[4] J.C. van Gemert, C.G.M. Snoek, C.J. Veenman, A.W.M. Smeulders, and J.M. Geusebroek, "Comparing compact codebooks for visual categorization," Comput. Vis. Image Understand., vol.114, no.4, pp.450–462, 2010.
[5] B. Ramamurthi and A. Gersho, "Classified vector quantization of images," IEEE Trans. Comm., vol.34, no.11, pp.1105–1115, 1986.
[6] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," Proc. CVPR, pp.2169–2178, 2006.
[7] A. Gersho and R.M. Gray, Vector quantization and signal compression, Kluwer Academic Publishers, Boston, 1992.
[8] F. Jurie and B. Triggs, "Creating efficient codebooks for visual recognition," Proc. ICCV, pp.604–610, 2005.
[9] J.C. van Gemert, C.J. Veenman, A.W.M. Smeulders, and J.M. Geusebroek, "Visual word ambiguity," IEEE Trans. Pattern Anal. Mach. Intell., vol.32, no.7, pp.1271–1283, 2010.