Image Categorization using Codebooks Built from Scored and Selected Local Features

Bala S. Divakaruni and Jie Zhou
Department of Computer Science, Northern Illinois University DeKalb IL USA 60115

Abstract: *This paper introduces a new method of codebook-based image categorization by building the codebook using scored and selected local features in the image. Different from traditional clustering-based codebook generation that may lead to codeword uncertainty and plausibility, the proposed Matching and Consensus (M&C) process follows the paradigm of feature selection: Based on distance metrics, the M&C process examines salient local features recurring over training images and produces scores that quantify the levels of relevance of the features to the image categories. By selecting features with the highest scores into the codebook, the method is expected to filter out non-representative candidates and keeps the most informative codewords for the category. We evaluate on five image sets for tasks of binary object identification and multi-class biological image classification. Experiments show that our method promotes very parsimonious codebooks that contain highly representative features and deliver a robust classification performance.*

Keywords: Object Recognition, Codebook, SIFT, Codewords, Classification and M & C based-scoring

1. Introduction

Our aim is to classify an entire image to a known category using sparse local features. In the past decade, opposed to methods using global descriptors, recognition based on local features including bags of local keypoints have been used widely in the classification of images or objects in images [3], [19]. In current literature, a codebook (sometimes called dictionary, vocabulary) is often generated to facilitate image representation and the subsequent classification. The most popular way to generate codes is to use clusters of the features in the training set. Images are then represented as histograms of these feature clusters. The basic clustering-based approach, however, has some known shortcomings including favouring dense regions and being non-discriminative that can lead to codeword uncertainty and plausibility [15]. Recent work introduced improvements such as assigning different weights to the generated codes [1], [15]. While some of them showed refined results on standard data sets, the focus has been put on weighting the codewords after they were already generated using the traditional codebook generation approach and large codebooks were used.

Differently, we propose a new codebook generation method that follows the paradigm of feature selection: we evaluate salient local features from training images and only the most representative ones are selected and put into the codebook. To assist the selection, a scoring mechanism, called M&C-scoring, is devised to measure the suitability of an individual feature as a codeword for a particular image class.

Our method can bring some major advantages: 1) The code book generated using our method contains scored codes. The scores are the natural results of the feature selection process, with each score corresponding to the level of relevance of the code to a particular class. By incorporating only the highly scored features, the selection process is expected to filter out non-representative features and avoid some problems of the classic codebook generation using clustering such as the inclusion of plausible codes. The scoring process we use is based on a combination of distance metrics similar as done in image retrieval. It circumvents the computational bottleneck in solving complex optimization problems in other related methods. 2) Because we work with local image features followed by a selection process, our method does not explicitly based on shapes and does not require aligning images or segmentation of objects. The inputs are entire images with both background and foreground. 3) Our method uses codebooks that are several magnitudes smaller than state-of-the-art methods (a few hundreds versus several thousands of codewords) which lead to better computational feasibility.

To evaluate our method, we test on five image datasets including standard natural scene datasets and a multi-class biological image set from a newly emerged domain of neurobiology. Tests on these datasets prove the effectiveness of our proposed method. Experiments show that our method promotes very parsimonious codebooks that contain representative features and deliver a robust performance.

2. Related work

Codebook based on bags of keypoints has been used in many recent applications of image classification and computer vision [1], [19]. In most cases, the codebook is generated using K-means clustering with the cluster centers being the codewords. Despite its popularity, some shortcomings are also well recognized in the community [1], [15]. For example, K-means does not select the most informative features. Instead, it tends to focus on high density areas of the feature space and starves low density ones. The discriminability of the codewords is thus compromised.

Some most recent work have attempted to overcome these problems: Soft assignment to weigh the codewords is investigated in [15] in order to address codeword *uncertainty* and *plausibility*. In [1], codewords were weighed using supervised distance metric learning. While they gave some refinements of the generated codes, these methods are still based on codes that were already obtained using K-means clustering. In other words, weighting only happens after the codebook was already generated. The traditional codebook generation approach brought in many codewords that were not contributing to the classification process, which leads to at least a waste of codebook entries and possibly a compromise of the potential discriminative power of the codebook representation. A different and conceptually more ideal alternative would be going back to the codebook generation process and avoid bringing in uncertain or plausible codewords into the codebook. Based on this rationale, our work in the paper uses a different strategy from previous works in that we employed the approach of selecting the local features following the feature selection paradigm. The goal is to find the most representative features. Those features are chosen by a scoring and selection process and put into our codebook.

The codeword binary weighting is also close to our work in terms of motivation. In [17], codewords were merged to reduce the size of a large codebook generated using K-means clustering. In [14], a regular lattice is used to discretize the feature space. Mean-shift is incorporated into clustering in [6] to overcome shortcomings of K-means and a supervised feature selection is done for improving speed (instead of accuracy). These methods tend to use a large codebook varying from thousands to 1 million code word entries. While the above methods are still based on clustering-generated codewords and the best performances reported are obtained by large codebooks, our approach aims to select a parsimonious set of code words that can effectively classify images.

Another related idea was to investigate features repeatedly recurring over training images by clustering features across different images. Those methods are commonly based on segmented training samples or make use of explicit shape-based model [2], [18]. Some work has also investigated the implicit shape model that is represented by large number of salient patches [9]. In this paper, local salient points are selected as the base of our codes. Scale-Invariant Feature Transform (SIFT) [10] is adopted as raw features. By using local order-less bag-of-features as the base, segmentation of the images is not needed by our method in this paper, nor the need of image alignment.

3. Methodology

Our codebook construction and image categorization processes can be summarized using Figure 1:

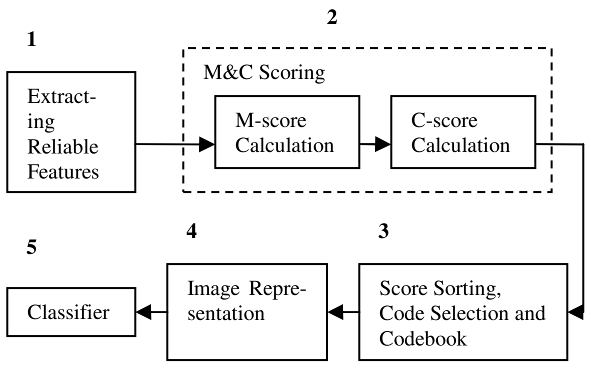

Figure 1: Diagram of the algorithm flow (codebook generation and training process).

As the first step, reliable local features need to be extracted from images. SIFT features [10] are assumed when explaining the algorithm, which are distinctive scale-invariant keypoints defined as maxima and minima of Difference of Gaussian (DoG) at various scales. However, any local keypoints can be used as candidate features for building codebooks using the proposed approach.

The features for training images then go through the M&C scoring process that selects the most suitable features (codewords) into the codebook, which is the focus of this section. The principle of the M&C scoring process is two-fold: Matching and Consensus. Matching is the process of finding similar features in different images of the same category. Consensus is the step based on matched features that investigates the frequency of recurring features in different images of the same category.

3.1. Matching Process and the M-score

The setting of the matching process is similar as in image retrieval, with features to be matched in the query image, and the remaining images viewed as reference images. The outcome of the Matching process is an M-score associated with each feature in the query image.

The M-score reflects the level of matching of the feature with different features in other images of the same category. As commonly used in image retrieval, we use distance-based metric to examine the level of similarity. Due to the fact that there are a large number of features in each image, we employ the Ratio of Distance (ROD) to improve robustness. ROD was defined the ratio of the closest distance (between the query feature and its best match) divided by the next closest distance (between the query feature and its second best match) when trying to match a feature with the features in another training image. ROD can be used to find a more robustly matched object [10], or a matched local feature based on the following rationale: a smaller ROD indicates that the best matched feature outperforms other close competitors, so the found match is more likely to be real instead of getting something from a random clutter.

Let I_i be the *i*-th image and the current query image, I_j being the *j*-th image that is viewed as a reference image. Say we have extracted m feature candidates from I_i, $\{f_i^1.., f_i^x \dots f_i^m\}$, and n features from I_j: $\{f_j^1.., f_j^y \dots f_j^n\}$, each is a local keypoint descriptor. Then Euclidean distance between two features are calculated and noted as $D(f_i^x, f_j^y)$ and are scaled between 0-1. For a given feature f_i^x in I_i, we look for its closest match in I_j, and we have

$$fd(f_{ij}^x) = \min\left\{D(f_i^x, f_j^y),\ y \in \{1,\dots n\}\right\} \qquad (1)$$

The Ratio of Distance is calculated as below:

$$rod(f_{ij}^x) = \frac{fd(f_{ij}^x)}{\min\left\{D(f_i^x, f_j^y), y \in \{1,\dots n\}\ \&\ D(f_i^x, f_j^y) > fd(f_{ij}^x)\right\}} \qquad (2)$$

where the denominator is the second closest match found in the image I_j for f_i^x.

We then define the M-score for the query feature f_i^x with respect to its match in I_j as:

$$M(f_{ij}^x) = 1 - fd(f_{ij}^x) * rod(f_{ij}^x) \qquad (3)$$

M-score is justified as following: The smaller the *fd*, the better the match; the smaller the *ROD*, the better the match. A match that is close (a smaller distance) and outstanding (a much better match than other candidates) gets a high M-score, which indicates an ideal match. To improve speed, some feature pairs with big *fd*s and big *rod*s were discarded early in the process without doing the calculation of M-score for f_i^x. Note that M-score is calculated when the feature is treated as a query feature so it is asymmetric.

3.2. Consensus Process and the C-score

Consensus is the process of determining how representative the matched features are for a particular class.

Due to intra-class variability, we do not always find matches for each feature in all the reference images even when we are examining the images labeled as the same category. For the purpose of selecting class-representative features, we look for the most frequently recurring features, i.e. the features with "consensus". Only features identified as containing matches in other images will be considered qualified to go through the consensus process. For this purpose, a threshold on M-score, *t*, is set on the M-scores to decide if we consider a match is found for a given feature x in the query image I_i when searching in image I_j. It is to help ensure the validity of matches while accommodating certain degree of intra-class variance. The consensus process calculates a C-score for a feature candidate x in image I_i as follows:

$$C(f_i^x) = \frac{1}{L-1} \sum_{j=1,,,i-1,i+1,,,L} \delta(M(f_{ij}^x) - t) M(f_{ij}^x) \qquad (4)$$

where L is the number of images for the given image category. $\delta(\cdot)$ is the indicator function that is 1 with positive input and 0 otherwise. A query feature can have a maximum of (L-1) matches so C-score is a number between [0, 1]. Intuitively, it can be understood as the consensus weighted by level of matching. Redundancy removal is incorporated in the process by keeping only one candidate in a matched set. This avoids repetitive features while reducing the computational cost.

M&C scoring process has associated a C-score with every qualified feature candidate. The generation of codebook is the matter of sorting of C-scores, then selecting those that with the highest scores. After M&C scoring process, top ranked features of each class are selected. They are combined to become the codebook for the given image classification problem.

With a generated cookbook, an image can be represented as a histogram of the codes, with the *i*th entry of a histogram being the proportion of features in the image having label *i* (i.e., the feature is closest to the code *i*). The task of image categorization is to apply the classifier to the represented images, either for the purpose of training or testing. We use support vector machine to accomplish the tasks. For the purpose of testing (i.e., categorizing unseen images), only steps 1, 4 and 5 in Figure 1 are needed.

As a summary, the M&C scoring process provides a quantitative measure of the level of suitability of codeword candidates, which in turn tends to avoid the codeword uncertainty and plausibility that were observed in traditional approach of codebook generation using K-means clustering. In addition, the score-based feature selection also brings a parsimonious codebook which will be further explained in experiments.

4. Experimental Results and Discussions

We test on the five datasets. The first set is K150, multi-class biological images containing 20 confocal microscope images of Drosophila (fruitfly) brain. It is a four category classification problem and each category represents a genetic line that highlights a subset of neurons (named a278, ato, a150 and a273). Image dimension is 998*530 with RGB channels. The green channel corresponds to the expressed neuron bundles and is used for categorization. The rest 4 sets are commonly tested object categorization datasets: Graz01 bike and people sets [13] and Caltech6 Motorcycle (side) and faces (front) sets [4]. They are binary problems with positive images containing the objects. Setups in [13] and [4] are used for Graz and Caltech6 respectively. Examples are shown in Fig. 2.

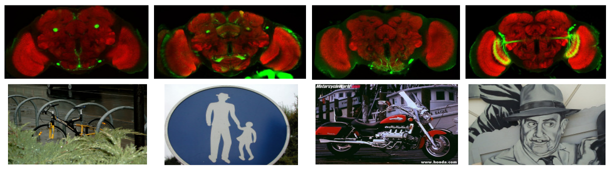

Figure 2: Image examples from K150 (top row, one image example per class) and positive images from other datasets.

4.1. Feature Preprocessing Using Hough Transform

The proposed M&C scoring works on individual features. In our experiments, we use local SIFT keypoint descriptors. However, a typical SIFT feature extractor results in hundreds to thousands features per image on the datasets to be tested. To reduce the computational complexity of feature scoring as well as the inference of some background clutters and outliers, we apply Hough transform [6] to the raw SIFT features. A raw SIFT feature extractor consists of 2D location, scale and orientation, and the keypoint descriptor. Hough transform is done for each image, using orientation, scale and 2D location as parameters for voting. In particular, we use 3 orientation bins, 2 scale bins and 16*16 location bins that are organized in a hierarchical fashion (each orientation bin has two scale sub-bins, each scale bin has 16*16 location sub-sub-bins). All bins are distributed equally in parameter space. Only the sub-sub-bins with sufficient entries of the features are kept, and those entries are identified as robust features. As the result of this step, the number of features per images is greatly reduced, with the resulting features being more robust candidates than raw SIFT features.

We need to point out that, while such voting involved in Hough transform may also be understood as clustering, the purpose and mechanism of Hough transform here is unrelated with K-mean clustering used in traditional code-book generation. First of all, Hough transform is done for each single image in our experiment for the purpose of removing outliers and obtaining robust feature candidates. It is a pre-processing step that is not concerned with features or codes from the entire set. It is the task of M&C scoring that selects and builds the codebook. Secondly, voting in the parameter space is done with equally distributed bins. This avoids the problems of concentrating on dense areas of the image of K-means clustering algorithm.

For all our experiments, each SIFT feature has a keypoint descriptor with the dimensionality of 128. We used JavaSift[1] and LibSVM[2] (linear kernel) for SIFT and SVM classifier. We implemented our algorithm in Java.

4.2. Experiments on Multi-class Bioimage Dataset

As seen from Figure 2, the set of images in K150 are pre-aligned with minimal background clutter. Due to its relative simplicity compared to natural scene datasets, it serves the similar purpose as a synthetic dataset in our project to demonstrate the discriminative capability of our proposed method. Meanwhile, it also demonstrates the method's effectiveness on a multi-class image set from biological domain.

1 http://pacific.mpi-cbg.de/cgi-bin/gitweb.cgi?p=mpicbg.git;a=summary

2 http://www.csie.ntu.edu.tw/~cjlin/libsvm/

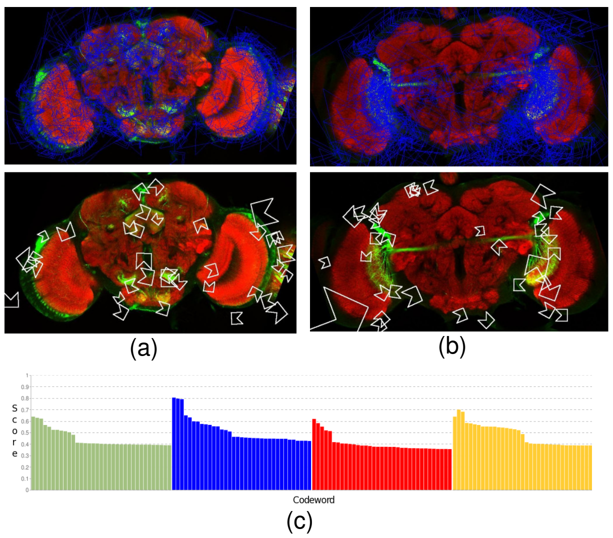

Figure 3: Examples of features and selected codes. (a) top: Raw SIFT features (blue); bottom: 35 selected codes for the class (white); (b) Raw SIFT feature for another image and 35 selected codes for the class. (c) Scores of the codewords in the codebook. Different colors indicate codewords for different classes.

The raw SIFT features and selected codes are visualized in Figure 3. From Fig. 3 (a)(b), we can see that the clutter background information from the original dense raw SIFT features were filtered after our codeword selection process. The selected features focus on the key structures of the gene expression patterns. Figure 3(c) shows scores of the selected codewords for the K150 dataset. In some previous work (e.g. [1]), many codewords created from traditional clustering carry zero weights after weighted based on their discriminative capability because those codewords were not useful for classification. Since our selection-based approach only keeps highly ranked codewords, it is apparent that we do not have such zero weight codewords. Including only useful codewords contributes to the parsimony of our codebook size. In fact, we only need 35 codes per class which is a total size of 140 entries to achieve a perfect recognition rate. The parsimony of our codebook will be further discussed in Section 4.3.

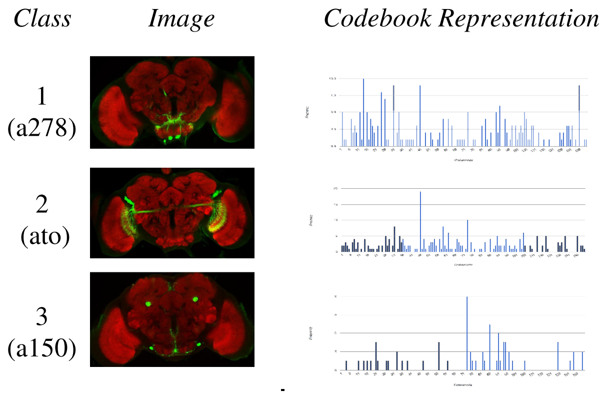

Figure 4: Histogram representation of images in K150. Codebook size is 35 codes per class. The codewords on the x-axis are numbered in the order of a278, ato, a150, and a273 from left to right. For example, the first 35 codewords belong to a278 (class 1) and so on.

In order to demonstrate the discriminative power of the selected code words, Figure 4 gives some examples of the histogram representations of images in K150. We can see that the codewords of corresponding category for the image show higher spikes (higher frequency) than other categories. These visually discriminative image representations are strong evidence that the selected codewords are class-informative.

Because the size of K150 set is small, we conducted Leave-one-out (LOO) cross validation in our experiment. LOO treats one image as the testing image while the remaining images are the training set, and repeats the experiment on all images. The matching threshold *t* is set to 0.6. With as few as 35 codes per class, our experiment on the 4-class K150 image set achieves 100 recognition rate, despite that there are only 5 images per class. (The class to which the testing image belongs has 4 training images.) The result is not sensitive to codebook size. Any codebook size between 30 and 45 yields around 95% accuracy (See Fig. 7(a)). This satisfactory result further confirms that representations based on our codebook capture the category-correlated properties of the image very well, as illustrated in Figure 4. K150 is a simple set compared with natural scenes with cluttered background. Nevertheless, the result shows the effectiveness of our model for extracting representative and discriminative local features as well as the model's applicability on classifying multi-class images.

4.3. Experiments on Binary Datasets

We then tested our algorithm on four standard object classification datasets. Examples of the Hough-transformed SIFT features are visualized in Figure 5. Figure 6 visualizes the results of selected codes using M&C scoring process. Note that different from k150 set, natural scene image sets are not aligned, so it is not straightforward to visualize the locations of selected codes from all training images on one particular image. In order to show the codes in an intuitive way, we picked a small set of images that are loosely aligned, and then did the selection on the aligned set. We used 2 images for Graz bike and Graz person, 6 images for Caltech Motorcycle, 8 for Caltech faces.

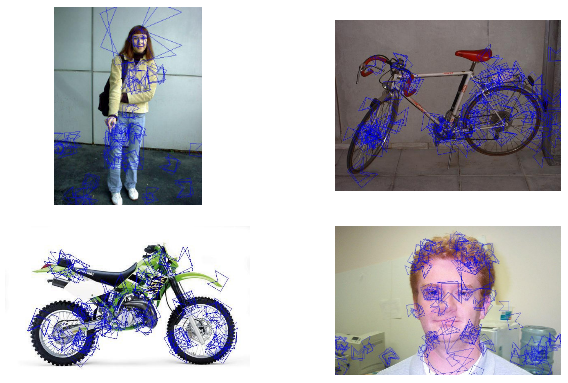

Figure 5: Examples of feature candidates extracted from image set (Hough-transformed SIFT features).

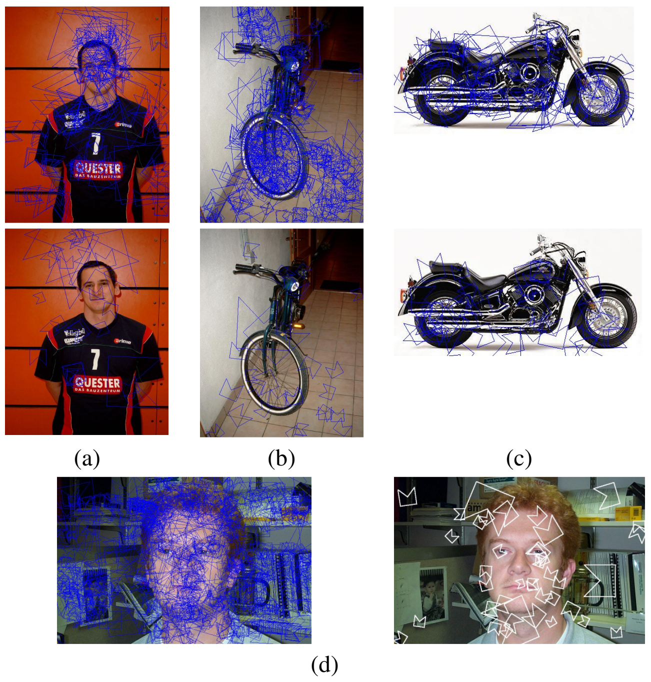

Figure 6: Use aligned images to visualize selected codes. (a)-(d): Graz people, Graz bike, Caltech Motorcycle and Caltech Faces, respectively. For each pair of images, one visualizes raw SIFT features (before applying Hough transform) and another contains top 40 selected features. This figure is better viewed in colors.

It is evident from Fig. 6 that dominant features are chosen as the result of our scoring and selection process. For example, in Caltech Motorcycle (c), features around the wheels and the engines are kept as the result of selection; while in Caltech Faces, selected features focus on the face prominent characteristics such as eyes, mouth and chin. The clutter background information from the original raw SIFT features were filtered out after our codeword selection process. We need to note here that since the training sets for the purpose of visualization are extremely small, some features that are common for the small aligned set would not be common for the actual object. For example, the plug-in on the wall was selected for the bike image because it appeared in both (aligned) training images. Such selection is less likely when more training images are used.

We report the classification accuracy in Table 1. Recent literature results are also shown as comparison. Table 2 lists the parameters used for the reported results.

Table 1: Experimental results.

Dataset	Graz Bike	Graz People	Caltech Motorcycle	Caltech Faces
Our result	**88.0**	**81.0**	**96.2**	**100**
[13]	86.5	80.8	94.3	100
[4]	-	-	92.5	96.3
[8]	86.3	82.3	-	-
[2]	79.0- 84.0	-	-	-

[19]	92.0	88.0	98.5	100
[1]	83.3-86.7	80.7-84.0	-	-

Table 2: Parameter Setting. *t*: matching threshold; Size: codebook size per class.

Parameter	Graz Bike	Graz People	Caltech Motorcycle	Caltech Faces
t	0.6	0.52	0.55	0.6
Size	112	167	424	40

From Table 1, we can see that, on three out of four datasets, our result improved the performance of [13] where SIFT feature and traditional codebook generation approach is used. On the remaining dataset Caltech 6 faces, our approach achieved the same very satisfactory result (100%) as [13] .

On Graz Bike, our result is also better than other state-of-the-art approaches reported in ([4], [8]). For example, we improved the result of [4] which obtained 79% to 84% depending on the complexity of the model, with 84% achieved by a complex spatial model. It is especially encouraging that on the versatile Graz 01 bike dataset, we outperformed the very recent result reported in [1] , where codes were first generated using traditional approach and then weighted using convex quadratic programming. On Caltech set, our results on both motorcycle and faces sets are very satisfactory, which are better than [4] in addition to surpassing or equal to [13] 's results. Overall, we obtained results that are better than or comparable to other recently reported models that we are aware of.

Our performance is lower than the best performance in [19] except that we match their performance on Caltech Faces, which is 100%. They made use of two feature detectors (Harris-Laplace and Laplace) and two descriptors (SPIN and SIFT). Given the conceptual simplicity of our model (and parsimonious codebook size as we will discuss below), we believe the experimental results successfully demonstrated the effectiveness and the discriminative capability of our model. In addition, while SIFT features are adopted in our experiment, other choice of local descriptors may be used for building a codebook using our M&C scoring process. Advanced combinations of descriptors such as those employed by [19] are interesting candidates for further exploration.

4.4. Codebook Size and Computational Complexity

Compared with the traditional approach of generating codebook, our approach has a much better flexibility in choosing codebook sizes, because the selection of codes was done after the scoring process is completed, while the number of codes (clusters) has to be preset in a clustering algorithm. This means that we can examine the effect of codebook size without major effort.

We have several interesting observations:
1. The codebook sizes we employed in our experiments are much smaller than those used in literature. For Caltech Faces and K150, 40 or less codes per class are needed to obtain the best results. For Graz Bike, 112 per class (224 codes total) are used. Comparatively, in [1] , the codebook size is 2000 for Graz dataset, which is about ten times larger than the size of our codebook. Due to the large codebook size, the authors had to put extensive effort to improve the efficiency when solving the optimization problem for calculating the codeword weights. Authors of [17] suggested codebooks greater than 3000 entries. In [14], the codebook size was as large as 5K or 10K; in [16], codebooks of 20K to 1M entries were experimented.

2. In general, we found that the best results of different data sets are obtained in different ranges of codebook sizes, possibly related to the complexity of the set. However, within that range, the performance of our method is stable. For example, the performances on k150 are all very satisfactory around the codebook size of 35 per class. Caltech Faces dataset has a very stable performance across almost any codebook size we tested round 40 per class. For the Caltech motorcycle dataset, while the best performance is 96.2% with 424 codes per class, codebooks sized between 300 and 700 per class reported very similar results around 95% with a variation of 1 to 2% (Fig. 7 (b)). In addition, the results are close with different matching threshold (*t*) within the range we tested ([0.5, 0.6]), although peaks may occur at slightly different values (See Table 2).

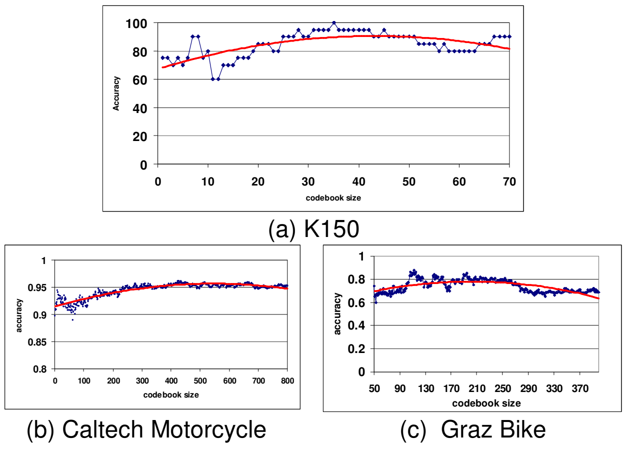

Figure 7: Trends of performance versus codebook size per class. Trend line (red) is fitted using polynomial degree 2.

3. We observe from trend lines in Figure 7 that when the codebook gets too large, the performance tends to drop. This observation is especially obvious on k150 and Graz Bike. We argue that the performance drop with large codebook is due to the inclusion of low ranked codes that are redundant or non-representative. The observation is consistent with general feature selection theory: Once a performance summit corresponding to an optimally selected set of features is reached, introducing more features will not necessarily increase the performance. Note that this observation is not the same as prior literature in which most studies tend to use larger codebooks. The size was restricted by computational

complexity but not by accuracy. For example, in [17], the accuracy-versus-size trend is reported to be monotonic, i.e. the bigger the size, the better the performance. A codebook size of greater than 3000 is thus suggested. While the optimal codebook size is likely to be application-related, we hypothesize that the monotonic trends reported in literature may indicate that the summit of the codebook's representative power is not yet reached even when the codebook size is already big, possibly related to the inclusion of non-contributing codes in traditional approaches.

The overall conclusion from the examination on codebook sizes shows that our method promotes parsimonious codebook that contains highly representative features and delivers a robust performance.

The parsimony of codebook size also contributes to computational feasibility of our method compared with other approaches. In the recent work of weighting the codes after they were generated from traditionally clustering approach, it typically takes hours for the weight optimization algorithm to converge, even after using some efficiency enhancement such as Alternating Optimization instead of Global Optimization [1]. In our case, the entire code generation process (M&C scoring and selection) takes about 30 minutes for the Graz datasets on a regular laptop computer (Intel Core 2 Duo CPU 2GHz, 3GB Memory, JVM 6.1). The analysis of time shows that most time of our algorithm is spent on the nearest neighbor matching process. Speed of the codebook generation and training process can be further enhanced if some improvement on nearest neighbor search is applied such as a k-d tree or its variants [12]. The testing process is not affected by the nearest neighbor matching and is very fast.

5. Conclusion

We proposed a way of building up codebooks for image classification using local feature scoring and selection. The scoring process is fundamentally built on top of a nearest neighbor matching by examining the sparse and robust local features repeatedly appearing in the training set and selecting the representative and recurring features. The effectiveness of such conceptually simple nearest neighbor approach has also been examined by related but different application of image annotation where the "strawman" approach of using distance metric to transfer labels was seen to outperform more complex parametric models on standard datasets for image annotation [11]. Our method, while with different purpose and setting, adds to the agreement of the importance of basic distance measure when incorporated into a suitable model.

6. Acknowledgements

We thank Julie Simpson at Howard Hughes Medical Institute for providing the bioimage datasets.

7. References

[1] H. Cai, F. Yan, K. Mikolajczyk, Learning Weights for Codebook in Image Classification and Retrieval, CVPR 2010.

[2] D. Crandall and D. Huttenlocher, Weakly supervised learning of part-based spatial models for visual recognition, in *Proc. ECCV*, pp. 16–29, 2006.

[3] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In Workshop of ECCV, pages 1-22, 2004.

[4] R. Fergus, P. Perona, and A. Zisserman, Object class recognition by unsupervised scale-invariant learning. CVPR, pp. 264–271. 2003

[5] V. Ferrari, F. Jurie, C. Schmid. From images to shape models for object detection, IJCV, 87(3):284-303, 2009

[6] E. Grimson. Object recognition by computer: the role of geometric constrains. The MIT Press: Cambridge, MA 1990

[7] F. Jurie and B. Trigger, Creating efficient codebooks for visual recognition, ICCV, pages 604-610, 2005.

[8] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories, CVPR, pages 2169–2178, 2006.

[9] B. Leibe, A. Leonardis, and B. Schiele, Combined object categorization and segmentation with an implicit shape model, in *Proc. ECCV Workshop on Statistical Learning in Computer Vision*, 2004.

[10] D. G. Lowe, Distinctive image features from scale-invariant keypoints, *IJCV,* 60, 2 (2004), pp. 91-110.

[11] A. Makadia, V. Pavlovic, and S. Kumar, A new Base line for image annotation, ECCV 2008.

[12] M Muja and D G. Lowe, Fast approximate nearest neighbors with automatic algorithm configuration, *Intl Conf on Computer Vision Theory and Applications,* 2009.

[13] A. Opelt, A. Pinz, M. Fussenegger, and P. Auer. Generic object recognition with boosting. IEEE Trans on PAMI, 28(3):416–431, 2006.

[14] T. Tuytelaars and C. Schmid. Vector quantizing feature space with a regular lattice. In ICCV pages 1-8 2007.

[15] J. C. van Gemert, J. M. Geusebroek, C. J. Veenman, and A. W. M. Smeulders. Kernel codebooks for scene categorization, In ECCV, 2008.

[16] J. Philbin, O. Chum, M. Israd, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. CVPR. 0:1-8, 2007.

[17] J. Winn, A. Criminisi and T. Minka, Object Categorization by Learned Universal Visual Dictionary, ICCV 2005

[18] J. Winn and N. Jojic, LOCUS: Learning Object Classes with Unsupervised Segmentation, ICCV, 756 – 763, 2005.

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