MULTIPLE DICTIONARIES FOR BAG OF WORDS LARGE SCALE IMAGE SEARCH

Mohamed Aly [1] *Mario Munich* [2] *Pietro Perona* [1]

[1] Computational Vision Lab, Caltech
Pasadena, CA, USA
{malaa, perona}@caltech.edu

[2] Evolution Robotics Inc.
Pasadena, CA, USA
mario@evolution.com

ABSTRACT

We explore building better dictionaries of visual words in Bag of Words (BoW) systems for searching large scale image collections. We propose a novel way to use more visual words in the BoW algorithm that significantly increases the recognition performance: combining multiple dictionaries built from different subsets of the features. We report improvement in performance by 40-45% over the baseline method and discuss how it can be parallelized on multiple machines to achieve faster run time. Finally, we show that our method is flexible, and provides 25-38% improvement over the start-of-the-art Hamming Embedding method.

Index Terms— Machine Vision, Image Retrieval, Object Recognition.

1. INTRODUCTION

Searching large scale collections of images has become an important application of machine vision. There are currently several smart phone applications that allow the user to take a photo and search a large database of stored images, e.g. Barnes and Noble[1] app. These image collections typically include images of book covers, DVD covers, retail products, and buildings and landmark images. They can grow to billions of objects, and thus typically contain one image per object. The ultimate goal is to identify the single database image containing the object depicted in a probe image e.g. finding a book given an image of that book cover from a different view point and scale. The correct image can then be presented to the user, together with some revenue generating information.

Bag of Words (BoW) approach has been shown shown to provide acceptable performance with fast run time and low storage requirements [1, 2, 3, 4, 5, 6, 7, 8]. It has been used in image retrieval settings [1, 2, 4, 8, 7, 9, 10], near duplicate detection [11, 12], and image clustering [5]. BoW has many parameters that affect its performance, including dictionary generation method, dictionary size, histogram weighting, normalization, and distance function.

In this paper we focus on the most important component: the dictionary of visual words. We present a novel method, Multiple Dictionaries for BoW (MDBoW), that uses more visual words (~ 5M) while significantly increasing the performance. Unlike previous approaches, we use more words from different independent dictionaries instead of adding more words to the same dictionary. The caveat is that the storage grows linearly with the number of dictionary used, however, with recent techniques for compact BoW [7, 9, 10], we can get improved performance using comparable storage. We also show significant performance gains by building the dictionary using all available features i.e. using features from *all* images indexed in the database. Finally, we report results significantly better than the state-of-the-art Hamming Embedding [4] method.

We make the following contributions:

1. We propose a novel method, MDBoW, that significantly improves the recognition performance of BoW large scale image search. We find performance improvements of 20-25% over the baseline and ~15% over the state-of-the-art.
2. We propose using features from all available images when building the dictionary. We report a performance gain of about 25%.
3. We show that combining these two ideas, we can improve the recognition performance by 40-45% over the baseline and 25-38% over the state-of-the-art Hamming Embedding [4] method.
4. We show that our method is easily parallelizable to achieve similar running times to using a single dictionary. Also, if combined with recent compact BoW methods, it could use comparable storage.

2. BAG OF WORDS (BOW)

BoW originated in text search applications [13] and has been successfully applied to various computer vision applications [1, 2, 3, 4, 5, 7, 9, 10, 14]. We focus on two variants of BoW search with Inverted Files (IF) [15]:

1. **Baseline IF**: this is the standard way of using IF with BoW. The dictionaries are built using Approximate K-Means (AKM) [2]. Histograms are normalized to have unit l_1 norm, then the l_1 distance is used to measure similarity between histograms [1, 16, 14].
2. **Hamming Embedding IF (HE)**: this is the method introduced by Jégou *et al.* [4]. The idea is to be more discriminative when computing distances between histograms. Instead of matching any two features that belong to the same visual word, each feature has an $N-$bit binary signature $b \in \{0,1\}^N$, such that two features f_i & f_j that belong to the same word will match only if the hamming distance $d_H(b_i, b_j) \leq T$ where T is some threshold. We use the settings from [4]: $N = 64$ bits, T=25, Tf-Idf histogram weighting, l_2 normalization, and the cos distance.

3. EXPERIMENTAL SETUP

3.1. Datasets

We use two of the datasets used in [14]. Please see Table 1 and refer to [14] for more details.

[1] www.barnesandnoble.com/iphone

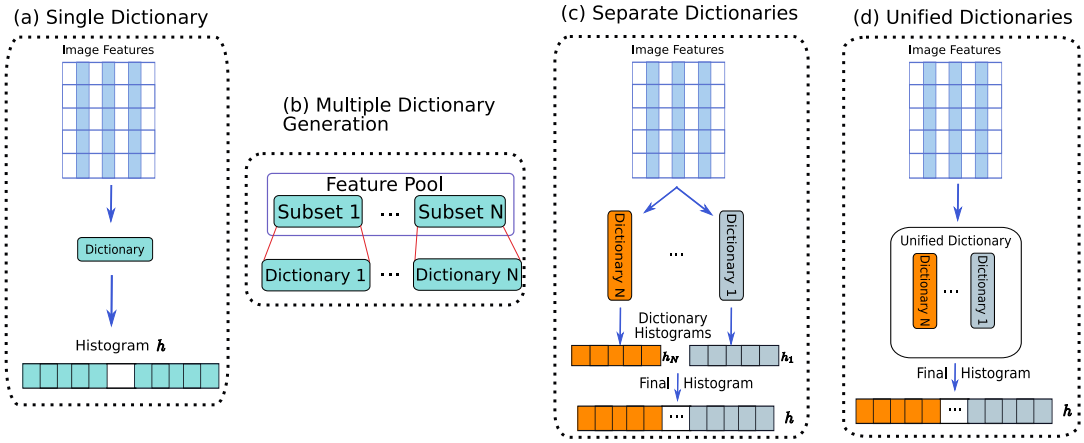

Fig. 1: **Schematic of the multiple dictionary generation**. **(a) Single Dictionary**: a single dictionary of visual words is generated from the pool of features, which is used to generate the histogram for the image. **(b)** Every dictionary D_n is generated with a different subset of the image features. **(c) Separate Dictionaries** (**SDs**): the image gets a histogram h_n from every dictionary D_n which are concatenated to form a single histogram h. Notice that every feature gets N entries in the histogram h, one from every dictionary. **(d) Unified Dictionaries** (**UDs**): a single unified dictionary is built from the concatenation of visual words from the dictionaries $1 \dots N$, and the image gets a single histogram h. Note that every feature gets only one entry in the histogram h.

3.2. Setup

We used two different evaluation sets, where in each we use a specific distractor/probe set pair, see Table 1. Evaluation was done by choosing 100K images from the distractor set in addition to all the model images from the probe set i.e. we have only one ground truth image per object. For every probe image (multiple per object), we get a ranked list of the images in the distractor + model sets, where highest ranked images are more likely to be the corresponding ground truth model image. Performance is measured as the percentage of probe images correctly matched to their ground truth model image i.e. whether the correct model image is the highest ranked image. All results reported in the paper are without any geometric consistency post-processing.

We want to emphasize that in our setup, we have only ONE correct ground truth image to be retrieved and several probe images. This setup is motivated by the application we consider: identifying the correct object in the database which has ONE image per object (e.g. searching for a book in a database of all book covers).

We use SIFT [17] feature descriptors (128 dimensions) with hessian affine [18] feature detectors. We used the publicly available binary [2] . All experiments were performed on machines with Intel dual Quad-Core Xeon E5420 2.5GHz processor and 32GB of RAM. We implemented all the algorithms using Matlab and Mex/C++ scripts.

[2] from http://www.robots.ox.ac.uk/~vgg/research/affine/

Probe Sets	
P1	Pasadena Buildings
P2	INRIA Holidays

Distractor Sets	
D1	Flickr-Buildings
D2	Flickr-Geo

Evaluation Sets		
Set	Distractor	Probe
1	D1	P1
2	D2	P2

Table 1: Distractor & Probe Sets and Evaluation Sets. See Sec. 3.1.

Algorithm 1 Multiple Dictionaries for Bag of Words (*MDBoW*)

1. Generate N random (possibly overlapping) subsets of the image features $\{S_n\}_1^N$
2. Compute a dictionary D_n independently for each subset S_n. Each dictionary D_n has a set of K_n visual words (cluster centers) $\{W_k^n\}_{k=1}^{K_n}$
3. Compute the histogram h for any image using the combination of the N dictionaries in one of two ways:
   (a) ***Separate Dictionaries*** **(SDs)**: For every image feature f_j, get its visual word w_j^n from every dictionary D_n. Accumulate these visual words into individual histograms h_n for each dictionary. The final histogram $h = [h_1^T \dots h_N^T]^T$ is the concatenation of the individual histograms.
   (b) ***Unified Dictionaries*** **(UDs)**: Combine the visual words $\{W_k^n\}$ from all the dictionaries into a unified set of words W. For every image feature f_j, get its visual word w_j from the unified set of words. Accumulate the visual words into the final histogram h.

The dictionaries were generated with random subsets of features of size 10M, out of ~150M features in the distractor set.

4. MULTIPLE DICTIONARIES FOR BOW (MDBOW)

We propose a novel way to increase the recognition performance of BoW: *Multiple Dictionaries for Bag of Words* (MDBoW). Our motivation is the view of BoW as an approximation to matching individual features [4] and the idea of Randomized Kd-Trees [17]. In this view, we are matching individual features using visual words as a proxy i.e. features that have the same visual word are considered matched. However, since we are dealing with very large dimensions, this approximation depends greatly on the random partitioning of the space offered by AKM. We can solve this problem by having mul-

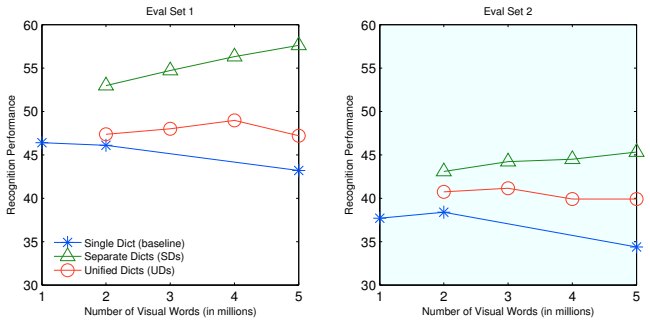

Fig. 2: **Multiple Dictionaries Results**. The X-axis shows the total number of visual words, while the Y-axis plots the recognition performance for 100K images using Baseline IF. For UDs & SDs, individual component dictionaries have 1M words each e.g. SDs with 2M visual words have 2 dictionaries. We notice a performance increase of about 25% using Separate Dictionaries with 5M words over the baseline of using 1M words. We also note that SDs have far superior performance than UDs. Note that simply increasing the number of words in a single dictionary (*blue*) is still worse than both SDs and UDs. See Sec. 4.

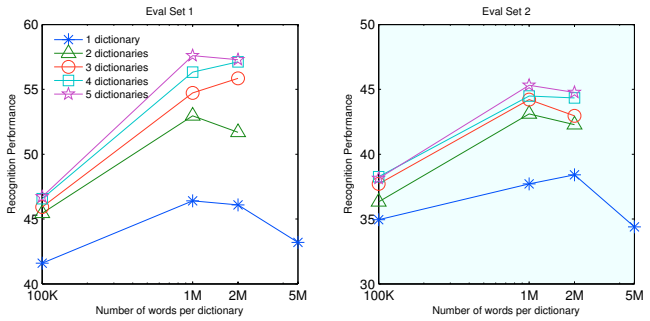

Fig. 3: **Parallelization of MDBoW**. The X-axis shows the number of visual words per dictionary, while the Y-axis plots the recognition performance for 100K images. Each point depicts the performance (on the Y-axis) of using a certain number of dictionaries (from 1 up to 5 dictionaries), one on each machine, each having the same number of words (on the X-axis). The best performance is achieved using 5 dictionaries (*magenta* curve) with 1M words each. Note that increasing the number of words in a single dictionary (*blue* curve) does not help beyond 1M or 2M words. See Sec. 4.

tiple independent dictionaries that give complementary partitions of the feature space, in the spirit of of Kd-Trees. Algorithm 1 outlines the idea, and please consult Fig. 1 for illustrations.

The advantage of SDs is that it is flexible and can be used with any kind of dictionary e.g. we can combine AKM and Hierarchical K-Means (HKM) dictionaries [1] with varying sizes. The drawback is that we are increasing the storage requirements in the inverted file, since each feature in the image has N entries in the final histogram, as well as the time to generate the visual words. On the other hand, UDs have the advantage of requiring less memory than SDs. However, these dictionaries must be of a type where it's easy to combine visual words from different dictionaries, which is not the case with HKM dictionaries for instance [1].

Fig. 2 shows results of using multiple dictionaries compared to using a single dictionary. All the dictionaries are generated with a random subset of 10 million features and use the baseline IF, see Sec. 2. We note the following:

- Using UDs (*red*) improves the performance slightly by about 3% over the baseline (*blue*).
- Using SDs (*green*) significantly improves the performance by about 20-25% over the baseline with 1 dictionary (*blue*). This can be explained by the fact that SDs use more information than UDs. In particular, every image feature in SDs gives information in each of the individual histograms, while in UDs that is not the case. This helps increase the performance, as we get multiple independent partitions of the feature space.
- Simply increasing the number of visual words in a single dictionary (*blue*) does not help, and in fact decreases the performance, as reported in [2].

We can view SDs as a trade-off between storage and computation required and recognition performance. Using more independent dictionaries enhances performance, but increases storage and computation, since we need to quantize more features and each feature contributes N entries in the final histogram (see Fig. 1). One way to solve this problem is to deploy each dictionary on a separate machine and then combine the results, see Fig. 3. The running time of the whole system would be the same as using one dictionary on a single machine, since they are running in parallel and we will have a significant increase in performance. We note that we can get the best performance when using 5 dictionaries/machines with 1M words each. Another way to solve the problem of increased storage is to apply SDs to one of the recent compact BoW methods [7, 9, 10] which have been shown to give comparable performance to the baseline IF with 1 dictionary while requiring a fraction of the storage e.g. 16-32 bytes per image.

5. MODEL FEATURES

The standard approach for building the dictionary of visual words has been to either use a set of unrelated images [2, 4] or to exclude the *model* images (ground truth images added to the distractors, see [14]) from this process [16, 14]. We argue that this is not necessary because in the actual setting, we have a set of images that we want to index (e.g. images of book covers). It would not make sense to build the dictionary on a separate set of unrelated images. Moreover, other image search approaches that are based on feature matching (e.g. Kd-trees) use all features available when matching the features of the probe image. That might be one reason why they have been reported to outperform BoW methods in this application [14].

Fig. 4 shows results of including the model features in the dictionary generation process. Dictionaries are generated with a random subset of 10M features with Baseline IF and 1M words, see Sec. 2. Probe images have on average 1K features each i.e. they constitute ~1% of the number of features used to build the dictionaries. We notice that adding all model features improves performance by up to 25% of the baseline.

6. PUTTING IT TOGETHER

Fig. 5 shows the results of combining the two ideas: using separate multiple dictionaries (SD MDBoW) (from Sec. 4) and including model features in dictionary generation (from Sec. 5). It compares both Baseline IF and Hamming Embedding (HE) IF [4], see Sec. 2. We notice that combining model features with MDBoW using either

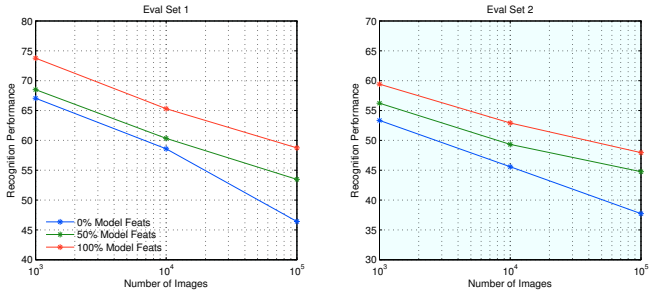

Fig. 4: **Model Features Results**. The X-axis shows the number distractor images in the database, while the Y-axis plots the recognition performance using Baseline IF. We notice a significant performance increase by including features of the model images in the dictionary generation. See Sec. 5.

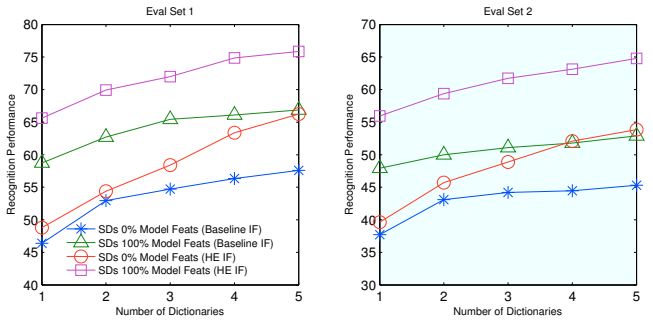

Fig. 5: **Combining Multiple Dictionaries (MDBoW) with Model Features**. The X-axis shows the number of dictionaries, while the Y-axis plots the recognition performance. We plot the baseline IF with/without model features and HE IF with/without model features. We notice about 40-45% increase in recognition performance over the baseline (1 dictionary) by combining model features and SDs (*green*) and about 20% increase over SDs alone (*blue*). We also notice that applying MDBoW to HE achieves 65-75% over the baseline and 25-38% over the state-of-the-art HE with 1 dictionary. See Fig. 2-4 and Sec. 4-5.

Baseline IF (*green*) or HE IF (*magenta*) significantly improves the performance over not using them (*blue* and *red*).

We note the following:

- For Baseline IF, using the model features with SD MDBoW (*green*) provides 18-25% performance increase over not using them (*blue*). It also gives a performance increase of about 40-45% over using 1 dictionary with Baseline IF.
- MDBoW with baseline IF (*green* at 5 dictionaries) significantly outperforms the sate-of-the-art HE IF with 1 dictionary (*red*) by ~15%.
- For Hamming Embedding IF, using MDBoW with HE without the model features (*red* at 5 dictionaries) gives a performance increase of ~32% over the state-of-the-art HE with 1 dictionary. Moreover, combining the model features with MDBoW using HE (*magenta*) gives a performance increase of about 65-75% over the Baseline IF with 1 dictionary and about 25-38% performance increase over HE with 1 dictionary.

7. SUMMARY

We explored ways to boost the performance of BoW image search methods by using more visual words. We presented a novel algorithm, MDBoW, and showed that using multiple independent dictionaries built from different subsets of the features increases significantly the recognition performance of BoW systems. We provided a simple way to parallelize N dictionaries over N machines and retain the run time of the baseline method. We showed performance improvements by 20-32% over the baseline and ~15% over the state-of-the-art. We argued that including features from indexed images when building the dictionaries is the right thing to do, and showed that it provides 25% improvement. We finally showed that combining these two ideas can yield 40-45% improvement in recognition performance over the baseline IF and 25-38% improvement over the state-of-the-art Hamming Embedding method.

Acknowledgements

This research was supported by ONR grant N00173-09-C-4005.

8. REFERENCES

[1] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," *CVPR*, 2006.

[2] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," *CVPR*, 2007.

[3] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman, "Total recall: Automatic query expansion with a generative feature model for object retrieval," in *ICCV*, 2007.

[4] H. Jégou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search," in *ECCV*, 2008.

[5] Mohamed Aly, Peter Welinder, Mario Munich, and Pietro Perona, "Towards Automated Large Scale Discovery of Image Families," in *CVPR Workshop on Internet Vision*, June 2009.

[6] O. Chum, M. Perdoch, and J. Matas, "Geometric min-hashing: Finding a (thick) needle in a haystack," in *CVPR*, 2009.

[7] H. Jégou, M. Douze, and C. Schmid, "Packing bag-of-features," in *ICCV*, sep 2009.

[8] Z. Wu, Q. Ke, M. Isard, and J. Sun, "Bundling features for large scale partial-duplicate web image search," in *CVPR*, 2009.

[9] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *CVPR*, 2010.

[10] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier, "Large-scale image retrieval with compressed fisher vectors," in *CVPR*, 2010.

[11] O. Chum, J. Philbin, M. Isard, and A. Zisserman, "Scalable near identical image and shot detection," in *CIVR*, 2007, pp. 549–556.

[12] O. Chum, J. Philbin, and A. Zisserman, "Near duplicate image detection: min-hash and tf-idf weighting," in *British Machine Vision Conference*, 2008.

[13] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, ACM Press, 1999.

[14] Mohamed Aly, Mario Munich, and Pietro Perona, "Indexing in large scale image collections: Scaling properties and benchmark," in *WACV*, 2011.

[15] J. Zobel and A. Moffat, "Inverted files for text search engines," *ACM Comput. Surv.*, 2006.

[16] Mohamed Aly, "Online Learning for Parameter Selection in Large Scale Image Search," in *CVPR Workshop OLCV*, June 2010.

[17] David Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, 2004.

[18] K. Mikolajczyk and C. Schmid, "Scale and affine invariant interest point detectors," *IJCV*, 2004.