A Multi-Sample, Multi-Tree Approach to Bag-of-Words Image Representation for Image Retrieval

Zhong Wu[*]
Tsinghua University

Qifa Ke[†1], Jian Sun[†2]
Microsoft Research
[1]Silicon Valley Lab, [2]Asia Lab

Heung-Yeung Shum[†]
Microsoft Corporation

Abstract

The state-of-the-art content based image retrieval systems has been significantly advanced by the introduction of SIFT features and the bag-of-words image representation. Converting an image into a bag-of-words, however, involves three non-trivial steps: feature detection, feature description, and feature quantization. At each of these steps, there is a significant amount of information lost, and the resulted visual words are often not discriminative enough for large scale image retrieval applications. In this paper, we propose a novel multi-sample multi-tree approach to computing the visual word codebook. By encoding more information of the original image feature, our approach generates a much more discriminative visual word codebook that is also efficient in terms of both computation and space consumption, without losing the original repeatability of the visual features. We evaluate our approach using both a ground-truth data set and a real-world large scale image database. Our results show that a significant improvement in both precision and recall can be achieved by using the codebook derived from our approach.

1. Introduction

SIFT features [7] and the bag-of-words image representation [19] are at the core of state-of-the-art large scale image retrieval systems. Converting an image into a bag of words involves three non-trivial steps: 1) feature detection, 2) feature description, and 3) feature quantization. Step 1 detects distinctive and repeatable image features such as DoG points and/or MSER regions [7, 8, 10]. In Step 2, an image patch for each feature is extracted, from which a feature descriptor is then computed [7, 9]. A quantizer in Step 3 is used to quantized a descriptor into a visual word in a pre-defined codebook [19]. Data structures like k-d trees,

vocabulary trees, or randomized forrest are often used in this step to speed up searching the best visual words in a codebook given a descriptor (c.f. [7, 12, 13, 15]).

Impressive results have been achieved for each of these steps in recent years. At each processing step, however, there is a significant amount of information lost, and the resulted visual words are often not discriminative enough for large scale image retrieval applications. Various approaches have been proposed to improve the discriminative power at different steps. At the feature detection step, multiple local features are grouped to form a more global and thus more discriminative feature (c.f. [1, 6, 16, 23, 24]. At the feature description step, higher-dimensional descriptors or descriptors augmented with other information [3, 9, 22] have been proposed to encode more image information. At the quantization step, approaches have focused on designing quantizers or codebooks that reduce quantization errors and/or preserve more information of the feature descriptor [4, 12, 15, 13, 18, 21].

We take a more global view of all the three steps, and propose a novel multi-sample multi-tree approach to computing the visual word codebook. Our goal is to increase the discriminative power of visual words by encoding more information of the original image feature. First, in Step 1, we sample multiple image patches at each image feature point/region, in contrast to just one image patch in previous approaches. Although these patches overlap with each other, when transformed to feature descriptors in Step 2, they encodes complementary information about the underlying image feature. In Step 3, we use multiple trees to quantize the descriptors; each tree representing a specific codebook is applied to one type of image patch sampling. The result is a packet of visual words for each feature point/region detected in Step 1. Each packet of visual words are then hashed into *one* final visual word in the codebook.

Our approach is novel in that it combines multi-sample features and quantizes them as a "visual packet" using multiple trees, which improves discriminative power. This is fundamentally different from previous approaches using multi-scale features [7, 5] and multiple trees [14, 11]. Our
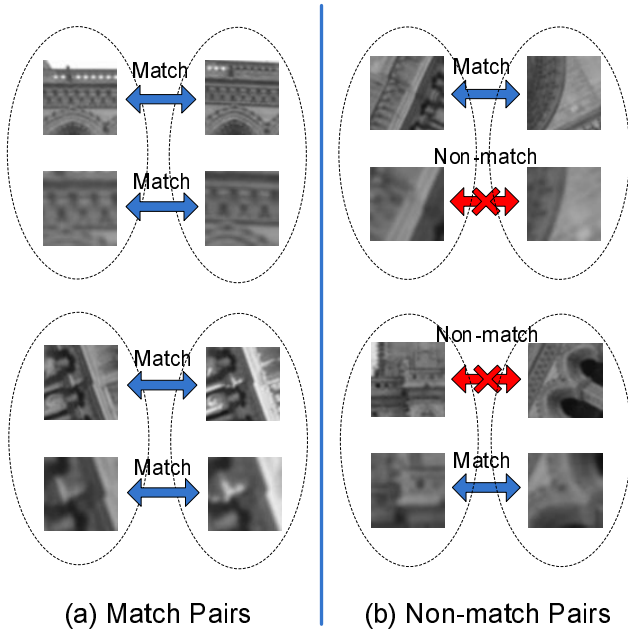
(a) Match Pairs   (b) Non-match Pairs

Figure 1. Examples of matched and non-matched visual packets. In each visual packet, two image patches are sampled at different scales. (a) two matched visual packets. In each compared visual packet pair, both large scale image patches and small scale image patches are quantized to the same visual word. (b) two non-matched visual packets. False matches can be effectively rejected since a visual packet is more discriminative than a single visual word.

approach has several advantages. First, by encoding more information about the underlying image feature using multiple samples and multiple trees, our approach generates a much more discriminative codebook. Second, since the image patch sampling process is deterministic, the repeatability of the original feature points/regions is also preserved. Third, by using multiple small trees, we derive a codebook that is not only more efficient (in terms of computation and space consumption), but also more discriminative than the codebook from a single tree that is an order of magnitude larger. The final result is a discriminative, repeatable, and efficient visual word codebook. We evaluate our approach using both a ground-truth data set and a real-world large scale image database, and show a significant improvement in both precision and recall in image retrieval.

2. Multi-Sample, Multi-Tree Representation

In this section, we first present the multi-sample, multi-tree approach to computing the bag-of-words image representation. We then show how to effectively use it in a large scale object retrieval system.

2.1. Representation - visual packet

Multi-sample. Let all detected feature points in an image $\mathbf{I}$ be $\mathbf{X} = \{x_i\}$. For each feature point, K image patches $\{p_i^1, p_i^2, ..., p_i^K\}$ are sampled. For example, we can sample image patches with different scales and locations around the feature point. Each sampled image patch p_i^k is then normalized to pre-defined image size (64×64), from which an 128-dimensional SIFT descriptor is then computed by assembling a 4×4 array of 8 orientated gradient histograms [7]. While different image patches may overlap in the image space, when normalized and transformed to SIFT descriptors, they effectively encode complementary information about the image feature. For example, a smaller patch sampled from a large patch, when normalized to 64×64, contains more fine grained and therefore higher frequency information.

Since we sample at pre-defined locations and scales, the repeatability of the underlying image feature is preserved. This is in contrast to approaches relying on grouping nearby features to improve discriminative power, where repeatability is decreased as it is non-trivial to group the *same* set of features at different imaging conditions.

Multi-tree. Each feature descriptor is quantized into a visual word d_i^k in a codebook which represented by a k-d tree. Instead of using a single codebook in the quantization, we use multiple trees and therefore multiple codebooks $\{\mathbf{B}^1, \mathbf{B}^2, ..., \mathbf{B}^K\}$ for the image patches obtained by different sampling methods. More specifically, the image patch p_i^k is quantized by the codebook $\mathbf{B}^k$. As a result, for each feature point, we obtain K visual words $\mathbf{d_i} = \{d_i^1, d_i^2, ..., d_i^K\}$. For simplicity, we call $\mathbf{d_i}$ a "visual packet", which is a multiple-sample, multiple-tree representation of the original image feature x_i.

An alternative to multi-tree approach is to concatenate all descriptors from multiple patches into one large descriptor. Such a higher-dimensional descriptor is also discriminative. One can then simply use one tree/codebook for quantization. This approach is not practical as it significantly increases the dimension of the feature space, making the k-nearest neighbor search of the visual words in the codebook exponentially more difficult and expensive.

One can also use one single tree to quantize all descriptors computed from all sample patches. To achieve similar discriminative power, the tree has to be significantly large to achieve a fine partition of the descriptor space, and is therefore more expensive in terms of both space and computation. We also found by experiments that a single giant tree achieves much worse precision and recall than multiple small trees. The reason is because different patch sampling methods lead to different descriptors that encode the image patch at different level of details. As a result, it is better to design one codebook for each patch sampling method.
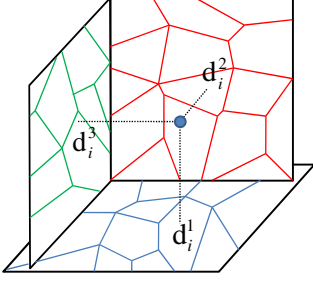
Figure 2. Partitions in the joint feature space. Each codebook represents a partition in a projected plane. Multiple codebooks construct a fine level partition of the joint space.

Different from the standard multi-scale sampling approach that detects features at different scales and then quantizes them separately, our approach combines multi-scale features sampled from the same feature point and then quantizes them into a "visual packet" using multiple trees. Note that multiple trees have been used to guarantee fast response in retrieval systems [14, 11]. Here we use multiple trees to construct a joint feature space by combining multi-scale features, thus to improve the discriminative power of visual words without losing repeatability.

Why more discriminative? A visual packet is more discriminative than a visual word because two visual packets are considered as identical only if their corresponding visual words are identical. Formally, it takes the intersection of the quantization results with multiple samples:

$$(\mathbf{d_i} = \mathbf{d_j}) \equiv (d_i^1 = d_j^1) \wedge (d_i^2 = d_j^2) \wedge ... \wedge (d_i^K = d_j^K). \quad (1)$$

Figure 1 shows several real examples of matched and non-matched visual packets. The false matches are effectively rejected because different samples can provide complementary information of a feature.

Geometrically, using the visual packet representation is equivalent to constructing a fine partition in the space of the joint feature $[p_i^1, p_i^2, ..., p_i^K]$. As illustrated by Figure 2, each codebook corresponds to a partition in a projected plane (a low-dimensional subspace) and multiple partitioned planes form a global and fine partition of the space of the joint feature. The number of partitions is up to N^K, where N is the number of visual words in each codebook.

2.2. Indexing

At a first glance, constructing the above large number of partitions in the joint-feature space is impractical in a large scale retrieval system. Fortunately, most of the partitions are empty given a limit number of images in the database. We can use the hash technique to efficiently convert a visual packet of a non-empty partition to a hash code. In our implementation, we use Rabin Hashing [17] $c_i = h([d_i^1 d_i^2 ... d_i^K])$, where $h()$ is the Rabin hash function
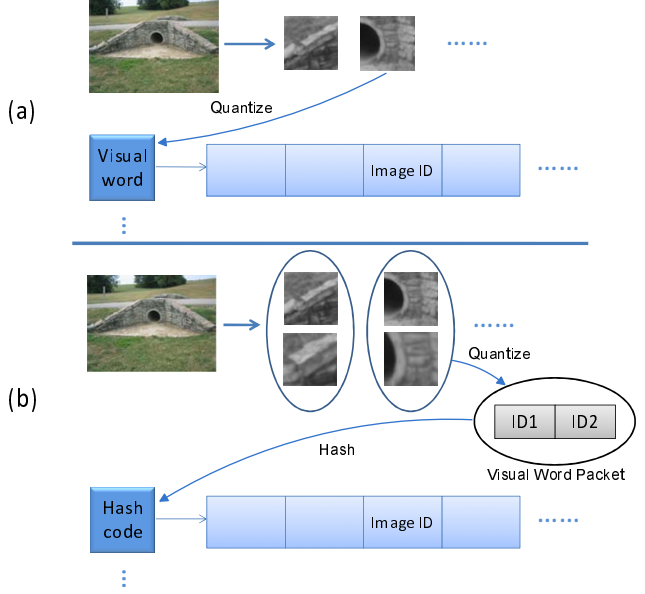

(a)

(b)

Figure 3. Inverted index structure. (a) using the standard visual word representation. (b) using the visual packet representation.

and c_i is the 32bits hash code. The Rabin hash function treats the concatenated visual words $[d_i^1 d_i^2 ... d_i^K]$ as a string and outputs an integer hashcode. The hash code plays the same role of the visual word in the inverted index file for image retrieval systems, as shown in Figure 3. In the standard visual words model, each feature is quantized to a visual word; in the visual packet model, each feature is quantized to a visual packet and then mapped to a hash code. For a given image database, we only need to store a set of small codebooks, a hash table, and a standard inverted index file. In our experiments, the size of hash table is often a fraction of the size of the inverted file. For small databases, it is about 50%–80% of the size of the index. With smaller codebooks and large databases, it is about 10%–30% of the size of the index.

2.3. Retrieval

Usually, increasing the discriminative power will inevitably decrease the recall of the retrieval system. To address this issue, we use soft quantization [15] in the retrieval stage to improve the recall. In soft quantization, each sampled image patch in the query image is associated with R nearest visual words in the codebook $\mathbf{B}^k$ instead of its single nearest visual word. Denote these R nearest neighbors as $\mathbf{S}_i^k = \{d_i^k(1), d_i^k(2), ..., d_i^k(R)\}$. Note that the soft quantization is only applied to the query image, not to the images being indexed. After the soft quantization, each detected feature will be associated with a set of candidate visual packets by enumerating all possible combinations:

$$\mathbf{d_i^{soft}} = \mathbf{S}_i^1 \times \mathbf{S}_i^2 \times ... \times \mathbf{S}_i^K, \quad (2)$$

where $\times$ is the Cartesian product. Although $\mathbf{d_i^{soft}}$ contains R^K visual packets, most of them have no matched entry in the hash table and are ignored. For example, the average number of matched visual packets is about $10-30$ when the soft quantization factor R is 40 in our experiments. Thus, the computation costs of the soft quantization for the standard visual words and the visual packets are comparable.

3. Experimental Results

In this section, we present our empirical studies of different sampling methods, and compare various methods for combine multiple samples for computing visual words.

We evaluate the performance of our multiple sample, multiple tree representation using two databases. One is the recent large image patch database [22] with known matching and non-matching image patch pairs. The matching pairs in this database are obtained by projecting 3D points from Photo Tourism [20] reconstructions back into the original images. With the matching information, we can accurately measure the performance at the patch level. The other data set is the object retrieval database [13] from University of Kentucky, which contains 10,200 images in sets of four images of one object/scene. Querying the database with each of the four images should ideally return the other three images in the set. Using this database, we evaluate the performance of our approach in an image retrieval system. We achieved similar performance gains using various features [22]. In this section, we report results on the widely used feature detector and descriptor– "DoG + SIFT" [7].

3.1. Results on image patch database

The image patch database consists of three datasets. Each dataset contains 100 thousand 64×64 gray-scale image patches. The Trevi Fountain (Roma) dataset is used as training data to learn visual word codebooks using standard k-means clustering algorithm. Another dataset, the Half Dome (Yosemite) dataset, is used as a test dataset for evaluation. We randomly sample 200K matched pairs and another 200K non-matched pairs from the test dataset for our evaluation.

For each pair of image patches, to simulate the indexing and retrieval phases, one patch is used in indexing phase and the other is used as a query patch. Note that soft quantization is only applied in the indexing phase. In other words, the indexed patch is quantized to its single nearest visual word in the visual word codebook. The query patch is soft quantized to its R nearest neighbors, i.e., quantized to R visual words. In our implementation of soft quantization, we use k-d tree [2] to find the R nearest neighbors of a given feature descriptor. We consider it a match if any of the R visual word is the same as the visual word of the indexed patch.
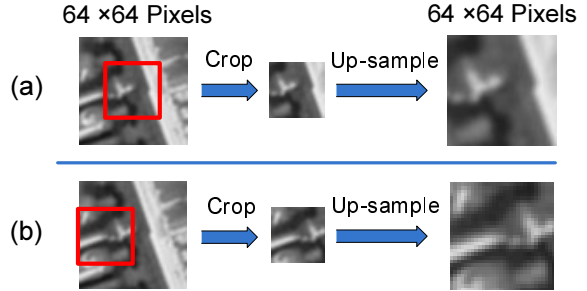

Figure 4. Patch sampling. (a) a central region is sampled and resized. (b) a non-central region is sampled and resized.
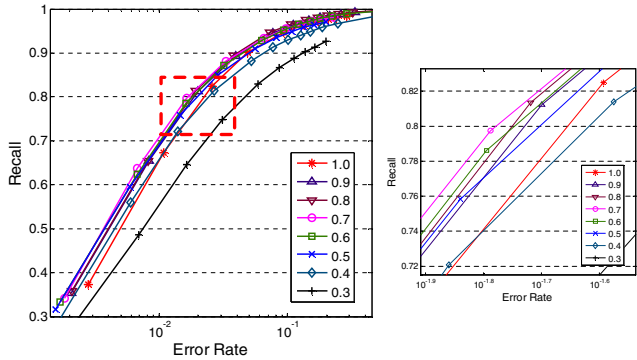

Figure 5. Recall-Error curves at different sampling scales. The X-axis is the error rate and Y-axis is the recall. The figure on the right side is a zoom-in view.

We measure the performance using *recall* and *error* metrics. The 200K matched pairs are used to measure the recall, and the 200K non-matched pairs are used to measure the error. The *recall* is defined as the fraction of the number of correct matches over the 200K true matches, and the *error* is the fraction of false positives, i.e., the number of false matches over the 200K non-matches. By varying the soft quantization factor R in the range {1, 4, 10, 20, 40, 60, 80, 100, 120, 150, 200, 500}, we can plot a "recall-error" curves as the overall quality measurement. While we use large quantization factors to perform a more exhaustive study, in real applications, a soft quantization factor between 2 and 40 is sufficient.

The feature detection step outputs a feature point/region together with a feature scale. This scale is usually multiplied by a const scale factor in order to extract an image patch for computing the feature descriptor.

Single scale sampling First, we study the impact of the sampling scale on the single image patch representation. Given a 64×64 image patch, we crop the central region and resize it to 64×64, as shown in Fig 4 (a). By varying the ratio of the size of the central region to the size of the original image patch, we essentially sample the image patch at different scales. The sampled (and resized) image patch is then transformed to SIFT descriptor and quantized to a
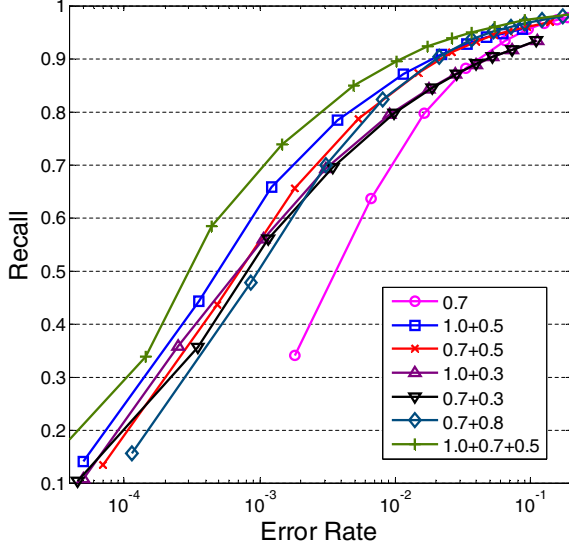
Figure 6. Comparison of the best single-scale sampling, several two-sample combinations, and the best triple-sample combinations.
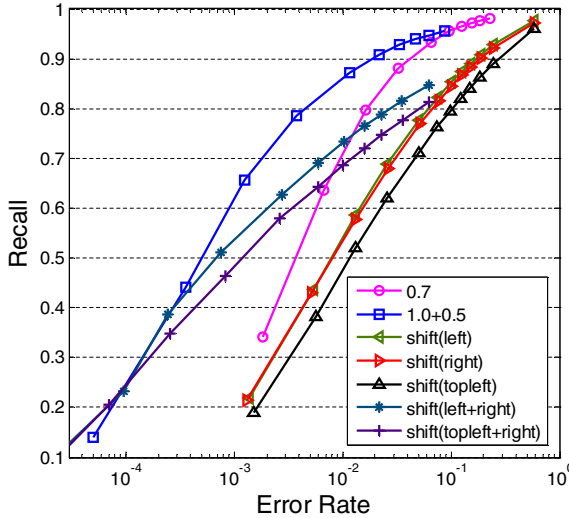

Figure 7. Recall-Error curves of using patches from different sampling approaches and their combinations. The X-axis is the error rate, and Y-axis is the recall.

Method \ Recall	0.4	0.6	0.8
1.0	3.10	7.70	22.1
0.9	2.60	6.40	18.7
0.8	2.50	6.30	17.8
0.7	2.40	**5.70**	**16.7**
0.6	**2.30**	6.00	17.9
0.5	**2.30**	6.00	19.8
0.4	2.80	7.30	24.1
0.3	4.20	12.8	46.6
1.0 + 0.8	0.60	1.60	6.20
1.0 + 0.5	**0.30**	**0.90**	**4.60**
1.0 + 0.3	**0.30**	1.50	9.70
0.8 + 0.5	0.40	1.20	5.20
0.7 + 0.5	0.39	1.30	6.40
0.7 + 0.3	0.53	1.60	10.3
0.7 + 0.8	0.46	1.70	6.80
0.5 + 0.3	0.60	2.50	14.4
1.0 + 0.7 + 0.5	**0.19**	0.50	2.90
1.0 + 0.8 + 0.5	0.20	**0.40**	**2.70**
1.0 + 0.5 + 0.3	0.20	0.70	4.90
1.0 + 0.8 + 0.3	0.20	0.60	4.70
0.8 + 0.5 + 0.3	0.20	0.80	5.40
shift(left)	4.30	14.3	62.5
shift(right)	4.30	15.4	67.2
shift(topleft)	6.30	22.4	103.2
shift(left + right)	0.28	2.00	27.6
shift(topleft + right)	0.44	3.40	50.7

Table 1. Error rates (10^{-3}) of different methods with respect to the recall.

Multiple scale sampling   Next, we exam the performance of the multiple-sample, multiple-tree representation. We perform an exhaustive search to find the best two-sample combination and triple-sample combination. Due to the space limit, we only show several representative combinations in Figure 6 and Table 1.

In Figure 6, we can see a substantial improvement from the best single sample "0.7" to the two-sample "1.0 + 0.5". At the recall 0.6, the error rate is reduced from 5.7×10^{-3} to 0.9×10^{-3}, which means that near 84% incorrect matches are rejected without sacrificing the number of correct matches! Using a triple-sample "$1.0 + 0.8 + 0.5$", we can further boost the performance at the cost of introducing an additional tree - the error rate is reduced to 0.4×10^{-3}.

More results shown in Table 1 lead to several observations:

- The combination of more complementary samples is better than the combination of less complementary samples. For example, the combination of "1.0 + 0.5"
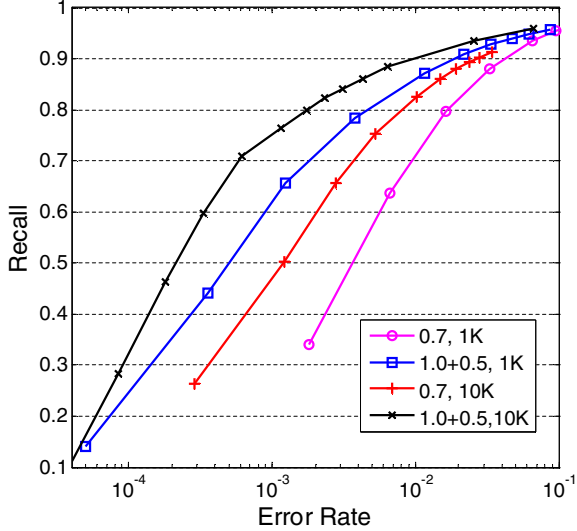
visual word.

Figure 5 plots the recall-error curves using different sampling scales. We can see that the performances of all single image patch representations are close. The result with sampling scale 0.3 is worst because the sampled image patch at this scale is too small to carry enough image information. On the other hand, the result with the largest sampling scale (1.0) is not the best since we use a fixed dimension descriptor. The best performance is achieved at the sampling scale 0.7, although its superiority is not significant.

Figure 8. Recall-Error curves from using different size of visual word codebooks. The X-axis is the error rate, and Y-axis is the recall. "1K" and "10K" indicate the size of visual word codebook.

	Time Cost	Recall	Error Rate
0.7, 10K	4.65	63.5	3.60
1.0 + 0.5, 1K	1.46	63.9	0.80

(a) Results using soft quantization factor 10.

	Time Cost	Recall	Error Rate
0.7, 10K	5.11	73.7	5.50
1.0 + 0.5, 1K	1.63	76.6	2.70

(b) Results using soft quantization factor 20.

Table 2. Comparisons of the quantization time cost (in milliseconds) per feature, recall (%), and error rate (10^{-3}). "1.0 + 0.5, 1K" is our multiple sample approach using two 1K visual word codebooks, and "0.7, 10K" is the single sample approach with a 10K codebook.

is better than other two-sample combinations, and the combination of "0.8 + 0.5" is better than that of "0.7 + 0.5".

- Combining a bad sample, for example scale "0.3", harms the performance.

- The combinations with more samples consistently achieve better performance. However, while 2-sample significantly outperforms 1-sample, the improvement of 3-sample over 2-sample is limit, indicating a saturation of performance improvement by combining more samples. Note that using more samples requires more k-d trees and more computations on the nearest neighbor search. Thus the 2-sample combination is a good trade-off between performance improvement and cost.

Multiple location sampling An alternative way to generating different samples is to sample regions at different locations instead of at the center, as shown in Fig 4 (b). However, due to the rotation and 3-D distortions, the non-central samples are less repeatable and lead to poor performances. In our experiments, we sample the patches to the left, right and top-left of the image patch center, and evaluate their performances. From Figure 5 and Table 1 we can see that they all achieve poor performances when compared to multi-scale sampling approach.

Small codebook V.S. large codebook To verify the consistency of the improvement given different codebook sizes, we compare the performances of four approaches: single sample 0.7 and two-sample $1.0 + 0.5$, using 1K and 10K visual word codebook respectively. From the results shown in Figure 8, we can see that the multiple-sample

representation achieves the same amount of significant improvements in both 1K case and 10K case. More surprisingly, the curve "1.0 + 0.5, 1K" is even better than the curve "1.0, 10K", while the former approach only uses two much smaller visual word codebooks. This is a nice property since using small codebooks is more efficient in terms of both storage and computation. Table 2 shows the time costs of four approaches. Compared to the single sample approach with 10K visual words, the multiple sample approach with two 1K visual-word codebooks not only achieves better performance in terms of recall and accuracy, but is also three times faster.

### 3.2. Results on object retrieval database

The University of Kentucky object retrieval database [13] originally contains about 10K images. We further add 100K images crawled from the Internet to form a large, real-world object retrieval database. Following [14, 4], we use mean Average Precision (mAP) as our evaluation metric. For each query image we compute its precision-recall curve, from which we obtain its average precision and then take the mean value over all queries.

We experimented with different sizes of visual word vocabulary, then chose the 100K visual word codebook and the soft quantization [14] with a factor of four as the baseline system. The features are detected by the DoG operator [7] and transformed to 128-dim SIFT descriptors. For each feature, we use its default scale estimated by the detector.

Since searching the best combination of samples on this large database is impractical, we simply choose a two-sample two-tree combination for evaluation and leave more comprehensive studies as the future work. The two-sample combination is: one is the original image patch and the other is the image patch at the same location but twice larger.
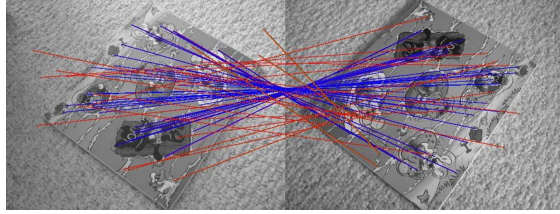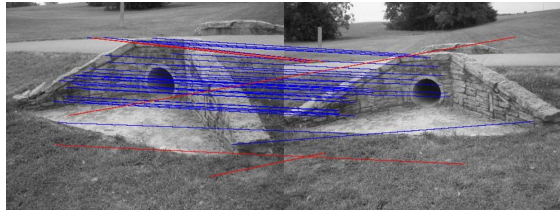
(a)   Correct: 48   Wrong: 20
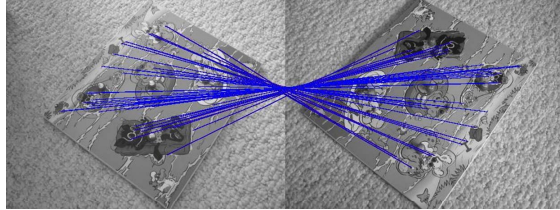
(b)   Correct: 69   Wrong: 0
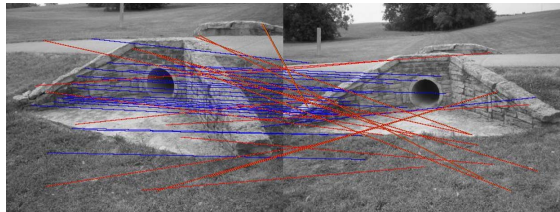
(c)   Correct: 24   Wrong: 19

(d)   Correct: 38   Wrong: 5

Figure 9. Image matching examples. Blue lines indicate the correct matches and red lines indicate the wrong matches. (a) and (c) are baseline results using single sample, single tree. (b) and (d) are our results using multiple-sample, multiple-tree representation.

We also experimented with the "multi-sample, single-tree" combination, which uses one vocabulary tree to quantize multi-scale features. The performance turns out to be similar to the baseline and is significantly inferior to two-sample two-tree combination.

Qualitative results. To visualize the improvement of our approach in object retrieval, we show two image matching results in Figure 9. The image on the left is the query image and the right one is the indexed image. We can clearly see that the multiple-sample, multiple-tree approach gives more inliers and less outliers, compared to the baseline ap-
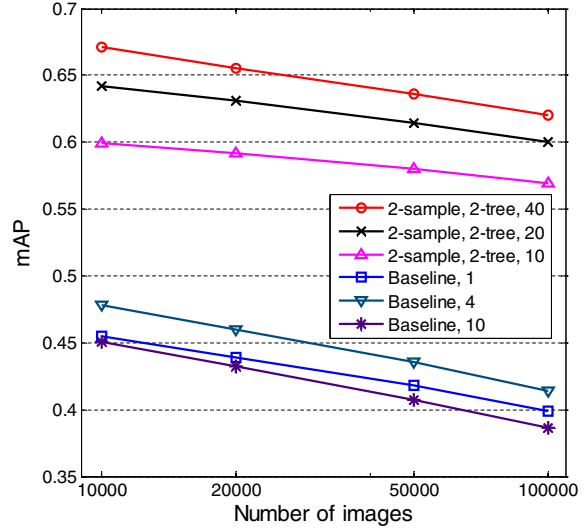

Figure 10. Comparisons of the baseline approach and our approach (2-sample 2-tree) using mAP, using different soft quantization factors. The numbers (following the name of the approach) in the legend are soft quantization factors.

proach.

Quantitative results. Figure 10 shows the quantitative comparisons using mAP by varying the number of images in the database. The smaller databases are obtained by sampling images from the largest database. Consistent with the results on the image patch database, the improvement by the multiple-sample, multiple-tree approach is also significant – about 40-50% improvements over the baseline approach.

Our approach also significantly outperforms the results reported in [13] using the metric of "top-4-score" [1]. On the 10K images database, our approach achieves a "top-4-score" of 3.60, while the best result in [13] is about 3.10.

To study the impact of soft quantization factor, we also vary the factor for both baseline approach and our approach. We can see from Figure 10 that a soft quantization factor of 4 works best for the baseline approach–increasing the factor to 10 results in a reduction in mAP. In our approach, we have found that increasing the soft quantization factor will improve the mAP. However, larger soft quantization factor incurs larger time cost. We choose the factor to be between 10 and 40 as the time cost is acceptable. In the following we present the runtime cost of our system.

Runtime. We perform our experiments with a single CPU on a 3.0GHz Core Duo desktop with 16G memory. Table 3 shows the runtime to query one image (excluding feature extraction time) on the 100K database, with different soft quantization factors used in Figure 10. We can see that with a soft factor of 10, our approach has a runtime cost slightly better than the baseline approach with a best soft

---

[1] The average number of true positives in the top-4 returned images.

factor of 4, while in this case our approach achieves a 39% percentages improvement in terms of mAP over the baseline approach with a best soft quantization factor. If higher time cost is allowed (i.e., 1.4 seconds query time, which is acceptable for many applications), we can achieve a 45% mAP improvement.

Soft factor	1	4	10
Time cost (s)	0.32	0.79	1.84

(a): Baseline approach.

Soft factor	10	20	40
Time cost (s)	0.71	1.39	4.50

(b): Our approach.

Table 3. Comparison of runtime per query image (without feature extraction) on the 100K image database using different soft quantization factors.

4. Conclusion

We have demonstrated that a simple and deterministic multi-sample multi-tree approach can effectively increase the discriminative power of visual word codebook, without losing repeatability of the original image features, yet efficient in terms of both computation and space consumption. We can consider the three steps for visual word computation as a pipeline consisting of three stages: 1) sampling image patches, 2) transforming image patches into feature descriptors, and 3) quantizing the descriptors into visual words. At each stage, different approaches can be applied to improve the discriminative power of the visual word codebook, by preserving more information about the original image patches. Taking one step further, an interesting problem is whether there exists an optimal codebook for bag-of-words image representation for the purpose of image retreival, and if so, how one can optimize the configuration of the pipeline in order to achieve the optimum. This is a complex but interesting problem that is useful for large scale image retrieval systems.

References

[1] A. Agarwal and B. Triggs. Hyperfeatures - multilevel local coding for visual recognition. In *ECCV*, 2006.

[2] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891–923, 1998.

[3] P.-E. Forssén and D. Lowe. Shape descriptors for maximally stable extremal regions. In *ICCV*, 2007.

[4] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *European Conference on Computer Vision*, 2008.

[5] S. C. R. B. Laptev I., Marszalek M. Learning realistic human actions from movies. In *CVPR*, 2008.

[6] D. Liu, G. Hua, P. Viola, and T. Chen. Integrated feature selection and higher-order spatial feature extraction for object categorization. In *CVPR*, 2008.

[7] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 20:91–110, 2003.

[8] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *BMVC*, 2002.

[9] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *PAMI*, 27(10):1615–1630, 2005.

[10] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *IJCV*, 65:43–72, 2005.

[11] U. H. Mikolajczyk K. Action recognition with motion-appearance vocabulary forest. In *CVPR*, 2008.

[12] F. Moosmann, B. Triggs, and F. Jurie. Randomized clustering forests for building fast and discriminative visual vocabularies. In *NIPS*, 2006.

[13] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *CVPR'2006*.

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

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

[16] T. Quack, V. Ferrari, B. Leibe, and L. J. V. Gool. Efficient mining of frequent and distinctive feature configurations. In *ICCV*, 2007.

[17] K. R.M. and R. M.O. Efficient randomized pattern-matching algorithms. *IBM J. Res. Dev.*, 31(2):249–260, 1987.

[18] G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *CVPR*, 2007.

[19] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, Oct. 2003.

[20] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. In *SIGGRAPH '06*, pages 835–846, 2006.

[21] T. Tuytelaars and C. Schmid. Vector quantizing feature space with a regular lattice. In *ICCV*, 2007.

[22] S. Winder and M. Brown. Learning local image descriptors. In *CVPR*, 2007.

[23] J. Yuan, Y. Wu, and M. Yang. Discovery of collocation patterns: from visual words to visual phrases. 2007.

[24] L. Zitnick, J. Sun, R. Szeliski, and S. Winder. Object instance recognition using triplets of feature symbols. TechReport MSR-TR-2007-53, Microsoft Research, 2007.