Towards Exhaustive Pairwise Matching in Large Image Collections

Kumar Srijan and C.V. Jawahar

Center for Visual Information Technology, IIIT, Hyderabad
kumar.srijan@research.iiit.ac.in, jawahar@iiit.ac.in
http://cvit.iiit.ac.in

Abstract. Exhaustive pairwise matching on large datasets presents serious practical challenges, and has mostly remained an unexplored domain. We make a step in this direction by demonstrating the feasibility of scalable indexing and fast retrieval of appearance and geometric information in images. We identify unification of database filtering and geometric verification steps as a key step for doing this. We devise a novel inverted indexing scheme, based on Bloom filters, to scalably index high order features extracted from pairs of nearby features. Unlike a conventional inverted index, we can adapt the size of the inverted index to maintain adequate sparsity of the posting lists. This ensures constant time query retrievals. We are thus able to implement an exhaustive pairwise matching scheme, with linear time complexity, using the 'query each image in turn' technique. We find the exhaustive nature of our approach to be very useful in mining small clusters of images, as demonstrated by a 73.2% recall on the UKBench dataset. In the Oxford Buildings dataset, we are able to discover all the query buildings. We also discover interesting overlapping images connecting distant images.

1 Introduction

The easy accessibility of large collections of images has opened opportunities for mining them. These datasets are excellent resources for location recognition[1], browsing[2], summarization[3], reconstruction[4] or creating walkthroughs[5] of various popular destinations. Given the current techniques for harvesting these large collections, they tend to be highly unordered and have a lot of irrelevant images. Hence, the automatic organization of these datasets is very much required. We present a method to organize these datasets as Image Match Graphs. This will allow the discovery of interesting intermediate images to connect distant images, as shown in Figure 1, and small clusters of matching images.

The match graph construction problem is to produce a graph denoting matches between any pair of images in the dataset. This problem is related to the image retrieval problem, where the user supplies a query image and the system returns a ranked list of similar images. Many of the recent techniques [6] for image retrieval utilize the Bag-of-Words(BoW) framework to implement a filtering stage whereby a large number of images are rejected. In this framework, visual

A. Fusiello et al. (Eds.): ECCV 2012 Ws/Demos, Part I, LNCS 7583, pp. 219–228, 2012.

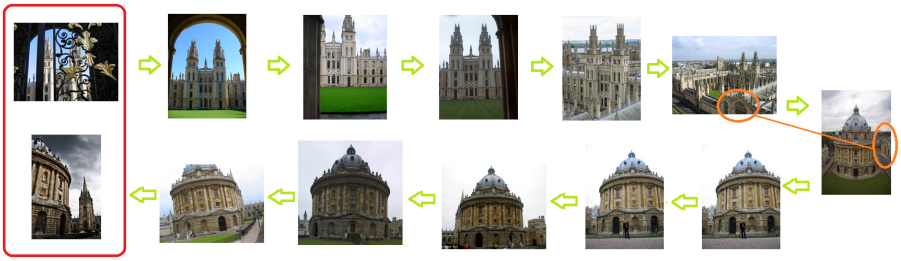


Fig. 1. A path discovered from an image of All Souls Building to an image of Radcliffe Camera in Oxford in our match graph

words are cluster centers extracted by clustering a large collection of descriptors extracted from various images. These visual words are used to quantize feature descriptors, such as SIFT [7], SURF [8] etc. To speedup the matching process, an inverted index is built which maps the visual words to a posting list of images which contain them. For every visual word in the query image, a vote is given to all the images which contain that visual word. A shortlist is obtained by taking into account the top scoring images. A geometric verification is performed on the images in the shortlist for reranking and rejecting non matching images.

Many of the recent techniques for image retrieval [9–11] try to bring geometry into the filtering stage to obtain more precise posting lists. Zhang et al. [9] use the geometry preserving visual words which captures both cooccurences, and local and long-range spatial layouts of the visual words to outperform even the BOW model using RANSAC for geometry verification. Similarly, [10] incorporate the neighborhood statistics of features into the vocabulary tree and in the spatial domain to improve the discriminative power of the features.

Efficient solutions for match graph construction also use techniques use for efficient image retrieval, such as feature quantization, indexing etc., to limit the amount of data that needs to be dealt with. Image retrieval also provides an immediate solution to the match graph construction problem: 'query each image in turn' and create a link from every image to each of its verified retrieved images [12]. Query expansion is used on these verified images to discover more overlapping images. Similarly, *Image webs* [2] use Image retrieval techniques to identify the skeleton of clusters, and complete the clusters by verifying potential links within a connected component with a focus on maximizing the *algebraic connectivity* of the cluster.

Chum et al. introduced Minhash based techniques [11, 13] which employ random sampling of the visual words using MinHash functions to obtain image signatures, similar to image histograms in the BoW framework. The signatures are sampled to obtain sketches, which become more discriminative than an individual visual word. Hence, all the sketch collisions, which are detected using a hash table, are verified. This makes the chance of discovery of a matching pair of images independent of the size of the database. The verified matches, called

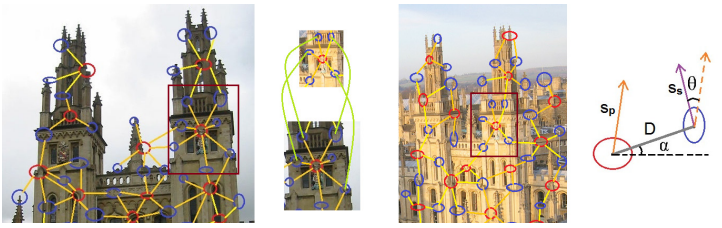


Fig. 2. A sampling of High Order Features(yellow) extracted in a pair of images. Geometric parameters(s_p/s_s , D/s_p , α and θ) are computed for a primary feature(red) with respect to all its secondary features(blue). s_p and s_s denote the scales of the primary and secondary features respectively. Vectors point towards the dominant orientation of the features. Four matching high order features are shown(green) which correspond to the primary features highlighted in the images.

seeds, are grown using query expansion to obtain full clusters. One disadvantage of this technique is that the chance of discovery of small clusters is not very high.

Notwithstanding, the success of the above techniques, it can be seen that the ideal solution for match graph construction is matching every image in the database to every other, that is, exhaustive pairwise matching. We explore the feasibility of doing this for building match graphs for large datasets. For this, we build upon the advantages of the above techniques: First, we employ an inverted index based retrieval scheme which provides direct access to the list of relevant images for a given query. Second, similar to the sketches used by Minhash based techniques, we use high ordered features, extracted from pairs of nearby features, to do feature matching in a more discriminative space. The indexing of geometry allows us to do match verification directly from index retrievals. We are thus able to implement 'query each image in turn' for exhaustive pairwise matching in linear time complexity.

2 High Order Features for Exhaustive Pairwise Matching

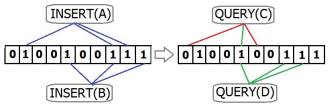
The direct implementation of 'query each image in turn' for building match graphs is not applicable due to its quadratic matching cost. In [12] and [2], the number of images needed to be taken into consideration for a query visual word is proportional to the size of the database. The number of such queries needed to be issued is proportional to the size of the database, making the overall timing complexity of the whole process quadratic in the number of images. Also, the need for keeping a shortlist of candidates from the filtering stage has the potential to miss out some of the matching images, thereby missing the quality of exhaustive pairwise matching. This issue becomes serious in the presence of a large number of distractor features, like those coming from trees, water etc., which dilute the contribution of visual words coming from the object in the image. This leads to relevant images not being able to make into the shortlist.

Min Hashing based techniques are able to bring down the computation cost by computing *sketches* which lead to lesser number of random matches than visual words. However, at a time, only one sketch per image is taken for matching as compared to a histogram level matching in [2, 12]. This affects the chance of discovery of matching images having only a few visual words in common. The chance of discovering a match also depends upon the size of the sketch. Choosing a low sketch size would find many matching images, but would also lead to many irrelevant sketch collisions in large datasets. In [13], using a sketch size of 3 for Oxford 100K Oxford Landmark database, 38.4 sketch collision were generated per image which lead to only 441 verified seeds in total. Therefore, a high sketch size is used in practice, but this leads to missing out seeds in smaller clusters and in clusters having low average image similarity. Moreover, since query expansion is used for completing the clusters, the success of these techniques is indirectly affected by the size of the database. The effect of these phenomenon is observed in the cluster representing the landmark "Magdalen Tower" in the Oxford Buildings Dataset [6], where only 3 of the 54 valid images were discovered, and no other image could be discovered through query expansion.

We identify scalable exhaustive pairwise matching as a feasible paradigm to overcome aforementioned issues. It can be seen that the inverted indexing technique could be made scalable if the size of the inverted index could grow with the size of the database, making the average size of the posting lists constant to allow quering in constant time. This is, however, not possible given the fixed domain of visual words. Therefore, we extract high order features by combining a feature with its nearby features and encoding their respective geometric configuration. This provides an extensive domain which is much more discriminative than visual words, and can be easily reprojected to a required size, using hash functions, based on the size of the database to obtain constant average size of the list. Zhang et al. have used a similar notion of high order spatial features in [14] to find all the cooccuring feature occurences under translation in a pair of images.

To address the problem of missing potential matches outside the shortlist, we design a match verification criteria which can be implemented on-the-fly from the index retrievals. This works well in practice as our high order features capture geometric information. Moreover, we match all high order features in a query image with all other high order features in the database, unlike [13], where only one sketch is pooled at a time per image for matching. This greatly enhances the chance of discovering small clusters.

Extracting High Order Features: We extract Hessian Affine regions and compute SIFT [7] descriptors of these regions for all the images in the database. These SIFT descriptors are quantized to a visual word vocabulary, using a kdtree built over the vocabulary. We choose nearby features for creating high order features, as their perspective projection into a matching image can be well modelled using a much simpler affine geometry. Next, we bin every image into bins of size 100 pixels. For each bin, we select upto 30 features, called *primary* features, by shortlisting features with the highest value of the scale parameter. Each *primary*



m/n	k=1	k=2	k=3	k=4	k=5
2	0.393	0.400			
3	0.283	0.237	0.253		
4	0.221	0.155	0.147	0.160	
5	0.181	0.109	0.092	0.092	0.101

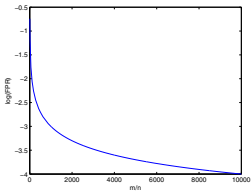


Fig. 3. (a)[left] The process of insertion and querying in a Bloom filter of size 10bit with 3 hash functions. [right] The first query is is not present in the Bloom filter. The second query is a false positive. (b) Table showing the False positive rates(FPR) for various combinations of parameters m(size of the Bloom filter)/n(number of elements to be indexed) and k(number of hash functions). (c) Table showing the variation of $log(FPR)$ with m/n for one hash function($k = 1$).

feature is paired with upto 20 of their nearest neighbours, within a radius of 80 pixels, to create high order features. Since our high order features are confined to local regions, we expect thier correspondences to follow affine geometry. Next, we compute the geometric parameters corresponding to the four grammar rules enlisted in [15]. Figure 2 gives a detailed description of these parameters. These grammar rules define invariants with respect to affine transformations. A high order feature is represented as a tuple enlisting visual words of the primary and the secondary feature, and the quantized values of the geometric parameters.

3 Indexing High Order Features Using Bloom Filters

The domain of high order features is obained by the cross product of the domain of visual words with itself and the domain of geometric parameters. Given a 1M vocabulary, the size of the domain of our high order features is 10^{12} even while neglecting the geometric parameters. This makes it possible to reproject our domain to a custom size, in accordance with the size of the database, to obtain sparse posting lists in the inverted index. To do this, we design an indexing scheme inspired from Bloom filters which minimize memory usage and provide constant time retrievals.

Bloom Filters: Bloom Filter [16] is a space efficient data structure for doing set membership queries. It allows constant time insertions and set membership queries. The downside of this scheme is that it occasionally identifies a non member query element as present. These are called false positives. Hence, the output for each query can be either “present in set, but can be wrong” or “certainly not in set”.

A Bloom filter is composed of a bit array, A, and a fixed set of associated hash functions, H. For inserting an element, e into the set S, all $h \in H$ are evaluated for e, and the bit positions corresponding to the resulting hash values are set to 1. For doing a membership query, $h \in H$ are evaluated on the query element, and all the bit positions corresponding to the resulting hash values are checked

for their set value. If any one of these bit positions is not set in the bit array, then the element is definitely not in the set, otherwise the element is deemed present. A false positive occurs when *all* the bit positions corresponding to the hash values of a query element have already been set by other elements inserted before.

Given the number of hash functions(k), the size of the bit vector(m) and the number of elements indexed(n), it is possible to determine the false positive rate by:

$$(1 - e^{-kn/m})^k$$

Figure 3 shows a simplified representation of Bloom filter operations, followed by a table showing false positive rates for low values of m/n and k. This is accompanied by a log plot of false positive rate with m/n for $k = 1$.

Indexing Using an Inverted Index over Bloom Filters: A simple indexing scheme can be built for N images, by allocating equally sized Bloom filters, $A_1 \ldots A_N$, with identical hash functions, and inserting the respective high order features of each of the N images. A query high order feature, q can now be resolved by evaluating all $h \in H$, and checking the corresponding bit positions in all the Bloom filters. It is easy to see that this process can be speeded up by storing the bit arrays of the Bloom filters as an inverted index over the bit positions. Retrieval can now be done easily by taking intersections of the posting lists of bit positions corresponding to evaluations of all $h \in H$ on q.

In the above framework, it is interesting to note that, keeping $k = 1$, that is, using only a single hash function, would eliminate any need of computing list intersections, making the retrieval process similar to that of BoW framework. However, unlike standard BoW, there is no restriction on the size of the inverted index, which is determined by the size of the bit array used. We can thus control the size of the inverted index in accordance to the size of the database to ensure a constant average length of the posting lists. This ensures constant time query retrievals. Only one hash function was also used by Mitzenmacher in [17] to make Bloom filters compressible.

One should, however, take care to always choose a big enough size of the inverted index so that the false positive rate while querying is low. Figure 3 shows the variation of false positive rate with m/n for $k = 1$. For a given maximum number, c, of high order features for any image in the database, it is easy to see that a false positive rate better than 10^{-3} can be easily achieved by allocating an inverted index of size $1000c$. Assuming 20k high order features in a query and database images, this implies generation of $20 \times 10^3 \times 10^{-3} = 20$ false matches on an average with every image.

Spatial Verification: We consider two primary features in different images a true correspondce only if they have at least v high order features originating from them in common. This criteria can be evaluated directly from inverted index retrievals, by querying all the high order features from a primary feature in succession and selecting images which claim to have v of these features. We consider an image level match verification to be passed if w such truly

Fig. 4. Two of the objects retrieved by our method for creating match graphs on the UKBench Dataset

corresponding primary features are found. This works well in practice as the high order features come from a very discriminative domain, resulting in very few mismatching of primary features across images. We keep the maximum number of secondary features per primary feature high at 20, to provide sufficient redundancy for finding common high order features with other primary features.

Given the susceptibility of our retrieval scheme to errors, it necessary for our spatial verification criteria to be robust to occasional mismatches. It is not difficult to see that our spatial verification scheme is robust to the introduction of a such few spurious mismatchings, because a single high order feature mismatch at random is not likely make the corresponding query primary feature truly correspond to another in a non matching image. Similarly, the event of multiple spurious high order mismatches corresponding to a single primary feature is also unlikely.

Match Graph Construction: We start by choosing an appropriate size, m, of the inverted index as discussed earlier. For Match Graph construction, we use a three-pass strategy, where we start by declaring a counter array of the size m to keep a count of high order features getting a certain hash value. In the first pass, we compute high order features and their corresponding hash value for all the images, while also updating the counter array. Now, precise memory allocations can be made for the posting lists of the inverted index based on the counter array. In the second pass, we index all the images by inserting their hash values computed earlier into the inverted index. In the final pass, we query each image in turn, and note down all the correspondences in an adjacency list.

4 Results

We use a standard image retrieval benchmark dataset, the University of Kentucky dataset(UKBench), introduced by Nister et al. [18], to measure our detection rate in small clusters. This dataset has 4 images each of 2550 objects making a total of 10200 medium resolution images. We used a vocabulary of 100K visual words for this experiment. On an average, 1048 features were extracted per image from this databset. An average of 413 primary features were selected for every image, which resulted in an average of 7436 high order features per image. This implies every primary feature combines with 18 secondary features on an average. A total of around 76m high order features were indexed using a simple FNV hash function. We defined the size of the inverted index to be 2^{25}, while the highest number of highorder features in a image was 23598. This corresponds to a maximum expected false positive rate of querying an image at $7{\times}10^{-4}$. This implies a maximum of 16.6 mismatches are expected to be generated between any pair of images.

The timing breakdown of the whole execution was as follows: Extraction the high order features and computation of their hash value took an average of 0.1 seconds per image. Building the inverted index over Bloom filters took 39 seconds. Querying the database took around 0.073 seconds per image. The total time required for the whole process was 23.6 minutes. The number of bytes required for indexing is equal to the sum of size of the inverted index and the number of high order features, that is, $8 \times 32 + 4 \times 76 = 560$Mb.

For the purpose of measuring our efficiency in detecting small clusters, we choose our match verification criteria as finding one primary feature with 3 high order features identical. We were able to find 1868 object clusters, corresponding to 73.2% recall, as compared to 49.6% recall reported in [13]. Figure 4 shows 2 of our retreived objects.

We have tested our approach on the challenging Oxford 5k dataset, introduced in [6]. It contains 5062 high resolution images of various buildings in Oxford, obtained by querying for building names on Flickr. Since, labels given to images tend not to be very accurate, this dataset contains a lot of distractors. Groundtruth is available for 11 of these buildings in the form of *Good*, *Ok*, and *Junk* images. Out of these *Good* and *Ok* images are considered true positives and the *Junk* images are considered as "don't care" samples.

For Oxford Buildings dataset, the maximum amount of high order features which got extracted for an image was 45k. The size of the inverted index used was 2^{29}. This gives a false positive rate of 8×10^{-5} for indexing 45k elements. This imples a maximum of 3.6 mismatches are expected to be generated between any pair of images. A total of 78 million high order features were extracted leading to a total memory requirement for indexing at $8 \times 512 + 4 \times 78 = 4408$Mb. The total time taken for extracting high order features and querying each image was $25 + 2 = 29$ minutes. For the high order features queried, the average length of the posting list was 1.16.

We kept the match verification criteria as finding atleast 3 primary features having atleast 4 high order features each identical between a query and a database image. We have computed the clusters as the connected component on the graph of matching images. In all 317 clusters were discovered containing a total 1367 images in them. The largest cluster has 362 images showing nearby All Souls building and Radcliffe Camera from various viewpoints. We were also able to find many interesting smaller clusters. These results are shown in Figure 6. We got mismatches, mostly in the form of text images, as shown in Figure 5.

We tested the scalability of our approach using the Oxford 105K dataset used in [13]. A total of 1480 million high order features were extracted. We used an inverted index of size 2^{29}, which resulted in an average length of posting list for the queried high order features at 6.8. The effect of larger average length of posting list can be seen at the average query time per image, which increased from 0.024 seconds to 0.086 seconds. A faster average query time can be obtained by using a larger inverted index. The total time taken for extracting features and querying was $9 + 2.5 = 11.5$ hours. The memory requirement for indexing was $8 \times 512 + 4 \times 1480 = 10016$Mb. We used the verification criteria as finding

Fig. 5. Text is is the most common source of errors in our scheme. In this particular case, a text image got matched to the window structure in the final image, which contains the landmark Radcliffe Camera. Hence, these images also become a part of the cluster containing Radcliffe Camera and All Souls Building.

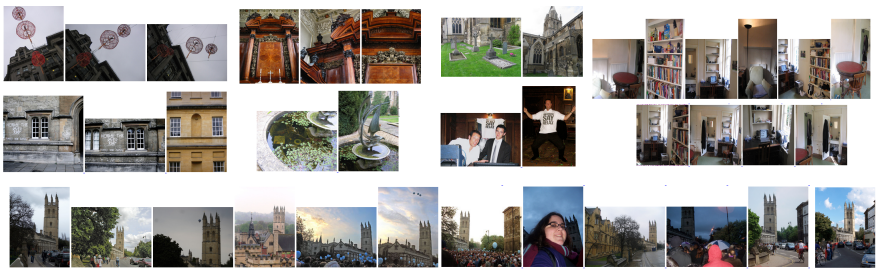


Fig. 6. Top two rows show small clusters identified by our method. Bottom row shows the cluster corresponding to 'difficult' Magdalen Tower.

atleast 6 primary features having atleast 4 high order features in common. We were able to obtain 2147 clusters involving 7198 images, with the largest cluster having 2265 images. Eyeballing this cluster revealed mismatches due to repeating patterns such as text, doors and windows which lead to coalescing of many smaller clusters.

5 Discussion and Conclusions

We show that it is feasible to index sufficient high order features capturing the appearance and geometric characteristics in an image, to an extent that there is no need for doing explicit geometrical verification. This is very advantageous as it eliminates any need for random disk accesses to fetch information required for doing geometrical verification. This is made possible as our geometric match verification criteria is computable directly from inverted index retrievals. We design an inverted indexing scheme which can adapt to the size of the database to ensure adequate sparsity of the posting lists to ensure constant time retrievals. The space savings are made by exploiting the behavior of an oversized Bloom filter using only one hash function. The extreme amounts of memory used by the Bloom filter is then shared efficiently using an inverted index structure for indexing multiple images. Our match verification is robust to the introduction of occasional spurious matches generated by our indexing scheme. Indexing high order features coming from an extensive domain would create problems for all the popular indexing schemes used for image retrieval or match graph construction, but we turn this

to our advantage, by devicing an indexing scheme based on Bloom filters, which uses constant storage per entry irrespective of the size or complexity of the entry.

In conclusion, our contributions can be summarized as: (i) introducing a novel indexing scheme suited for indexing and querying the geometrical and appearance information in images (ii) implementing an exhaustive pairwise matching scheme to build an image match graph for moderately large datasets in linear time (iii) introducing a geometric verification criteria verifiable at index.

Acknowledgement. This work is supported by the Department of Science and Technology, Government of India.

References

1. Li, Y., Snavely, N., Huttenlocher, D.P.: Location Recognition Using Prioritized Feature Matching. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part II. LNCS, vol. 6312, pp. 791–804. Springer, Heidelberg (2010)
2. Heath, K., Gelfand, N., Ovsjanikov, M., Aanjaneya, M., Guibas, L.J.: Image webs: Computing and exploiting connectivity in image collections. In: CVPR, pp. 3432–3439 (2010)
3. Simon, I., Snavely, N., Seitz, S.M.: Scene summarization for online image collections. In: ICCV, pp. 1–8 (2007)
4. Snavely, N., Seitz, S.M., Szeliski, R.: Photo tourism: exploring photo collections in 3d. ACM Trans. Graph. 25(3), 835–846 (2006)
5. Srijan, K., Ishtiaque, S.A., Sinha, S., Jawahar, C.V.: Image-based walkthroughs from incremental and partial scene reconstructions. In: BMVC (2010)
6. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: CVPR (2007)
7. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)
8. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: Speeded Up Robust Features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006)
9. Zhang, Y., Jia, Z., Chen, T.: Image retrieval with geometry-preserving visual phrases. In: CVPR, pp. 809–816 (2011)
10. Wang, X., Yang, M., Cour, T., Zhu, S., Yu, K., Han, T.X.: Contextual weighting for vocabulary tree based image retrieval. In: ICCV, pp. 209–216 (2011)
11. Chum, O., Perdoch, M., Matas, J.: Geometric min-hashing: Finding a (thick) needle in a haystack. In: CVPR, pp. 17–24 (2009)
12. Philbin, J., Zisserman, A.: Object mining using a matching graph on very large image collections. In: ICVGIP, pp. 738–745 (2008)
13. Chum, O., Matas, J.: Large-scale discovery of spatially related images. IEEE Trans. Pattern Anal. Mach. Intell. 32(2), 371–377 (2010)
14. Zhang, Y., Chen, T.: Efficient kernels for identifying unbounded-order spatial features. In: CVPR, pp. 1762–1769 (2009)
15. Xu, Y., Madison, R.: Robust object recognition using a cascade of geometric consistency filters. In: AIPR 2009, pp. 1–8 (2009)
16. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM 13(7), 422–426 (1970)
17. Mitzenmacher, M.: Compressed bloom filters. IEEE/ACM Trans. Netw. 10(5), 604–612 (2002)
18. Nistér, D., Stewénius, H.: Scalable recognition with a vocabulary tree. In: CVPR (2), pp. 2161–2168 (2006)