A Binning Scheme for Fast Hard Drive Based Image Search

Friedrich Fraundorfer Henrik Stewénius*
David Nistér†
Center for Visualization and Virtual Environments
Department of Computer Science, University of Kentucky
{fraundorfer,stewe,dnister}@vis.uky.edu

Abstract

In this paper we investigate how to scale a content based image retrieval approach beyond the RAM limits of a single computer and to make use of its hard drive to store the feature database. The feature vectors describing the images in the database are binned in multiple independent ways. Each bin contains images similar to a representative prototype. Each binning is considered through two stages of processing. First, the prototype closest to the query is found. Second, the bin corresponding to the closest prototype is fetched from disk and searched completely. The query process is repeatedly performing these two stages, each time with a binning independent of the previous ones. The scheme cuts down the hard drive access significantly and results in a major speed up. An experimental comparison between the binning scheme and a raw search shows competitive retrieval quality.

1. Introduction

Recently, [9] showed that efficient indexing and retrieval is possible with larger image databases than previously attempted. However, databases that contain 1 million images or more are too large to be kept in RAM of most computers. Such databases will have to either be handled by very large computer farms, or kept on hard drives where the access to the image data is slow.

In this paper we investigate an approach aiming to minimize the hard drive access and thus creating a fast hard drive based image search. Features are extracted from each image and quantized into an integer description as described in [9]. Inspired by the success of locality sensitive hashing for nearest neighbor search [3], we use multiple independent binnings of the database. Each binning is defined by a number of prototypes where a prototype is a vector representing an image. The images are assigned to the bin corresponding to the closest prototype, which is used as a proxy in the search. During a query we consider multiple such binnings, and each binning requires two processing stages. The first stage is to find the prototype closest to the query

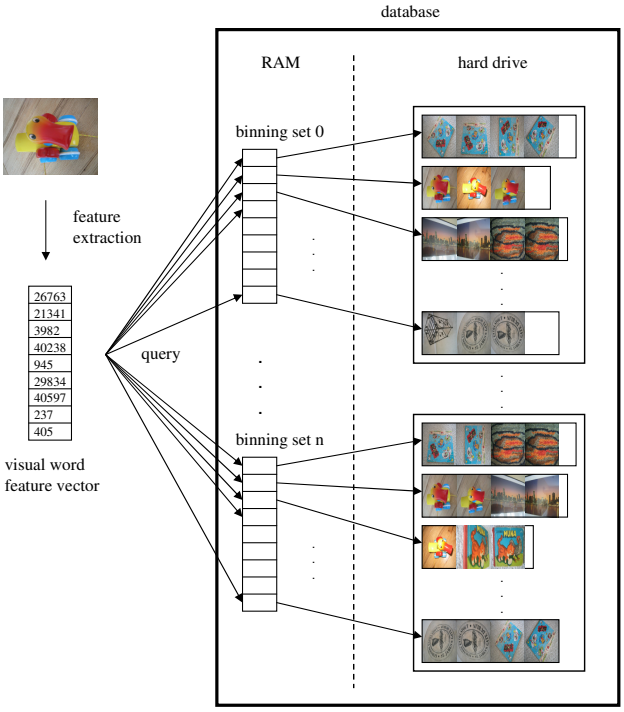

Figure 1. The proposed binning scheme for fast hard drive based image search. The feature vectors are stored on the hard drive in multiple independent binnings. The binning is defined by prototype vectors that can be kept in RAM. For an image query a visual word feature description is first extracted. Then all prototypes are searched to find the closest to the query vector for each binning. The associated bin is then fetched from disk and all images in it are searched. The final query result is the best match found over all binnings.

*Henrik Stewénius is now with Google Switzerland, Zürich
†David Nistér is now with Microsoft Live Labs, Redmond

1-4244-1180-7/07/$25.00 ©2007 IEEE

image. The second stage is fetching the bin from disk and searching through all the images in the bin of that protoype. The final result is the best image found over all binnings.

While the multiple binnings require more disk space, the use of protoypes brings a major speed up. An additional benefit of this scheme is that it is possible to trade retrieval quality for speed, so that in principle almost any retrieval time could be achieved, although the quality would eventually suffer.

In our experiments we will investigate how much the splitting of the database influences the retrieval quality. The retrieval quality of the binning scheme is compared to a raw inverted file approach and the results show that with multiple binning sets the quality of the raw inverted file approach can be matched.

2. Related work and state-of-the-art

A variety of authors [6, 11, 9, 10, 4] recently presented well working methods for image retrieval and object recognition using local features [8]. Enabling real-time image retrieval for larger and larger databases is now the focus of many research groups. The recognition approach of Lowe [6] can cope with around 5000 images in real-time. The images are represented as SIFT feature vectors [6] and recognition is done by nearest neighbor search in feature space. A k-d tree with a best-bin-first modification is used for efficient approximate nearest neighbor search. In [10] Obdrzalek and Matas present a method for real-time recognition of about 100-1000 objects. They use a decision tree for sub-linear indexing. A very fast approach is also presented by Lepetit *et al*. [4]. By using a randomized tree for keypoint indexing they achieve real-time object recognition and tracking. A large scale video indexing scheme was demonstrated by Sivic and Zisserman [11]. Their system allows a search for movie keyframes at interactive rate for feature-length movies. However, the current state-of-the-art is represented by the work of Nistér and Stewénius [9]. Their real-time demo allows object recognition from a database of 50000 CD-covers while their recognition scheme is scalable up to much larger databases. The same image representation and indexing scheme is used throughout this work. For an efficient database access scheme we adopt ideas from locality sensitive hashing for approximate nearest neighbor search as described in [1, 3]. Locality sensitive hashing has already been used for image retrieval by Grauman in [2].

3. Recognition with the vocabulary tree

We will give a short outline of the object recognition method that is also described in [9] to introduce the terminology which is used in this paper. Object recognition proceeds by a local approach, by detecting local features, computing a description (feature) vector and matching with a database of feature vectors. Each local detection (from the MSER detector [7]) is described by a SIFT feature vector [5]. Each SIFT feature vector is then quantized with the so-called vocabulary tree. It assigns a single integer value, denoted visual word (VW) to a SIFT feature vector. This eases matching a lot. Instead of computing distances between SIFT feature vectors only integer values have to be compared. Each image is then represented as a set of visual words. An image query is performed using a weighted voting scheme. An inverted file is created from a set of database images. For each VW the inverted file stores the database images where it occurs as an image list. When querying, the image lists for all VW's that occur in the query image are processed and a weighted vote is added to the images in the list. The database image with the highest score is selected as the closest match.

nr. images	inv. file size	read size	disk time
10000	40MB	200kB	10.0005s
50000	200MB	1MB	10.005s
100000	400MB	2MB	10.05s
1 million	4GB	20MB	10.5s
10 millions	40GB	200MB	15s
100 millions	400GB	2GB	60s
1 billion	4TB	20GB	510s
2 billions	8TB	40GB	1010s

Table 1. Specifics of a hard drive based inverted file approach. The table shows the size of the inverted file database, the data to read per query and the projected disk time per query for databases of 10000 to 2 billion images.

3.1. Recognition from the hard drive

For a certain number of images the inverted file is small enough to live in RAM. But unless very large computer farming is used, it has to be kept on disk at some point. The size of the inverted file (in Bytes) is calculated as

$$DB_{inv} = 4fI, \tag{1}$$

where f is the number of visual words per image, I is the number of images in the database and the factor 4 accounts for the use of 32 bit integers. The second column of Table 1 shows the database sizes for different numbers of images.

When scoring a query image only a part of the database has to be accessed and processed. The average amount of data (in Bytes) to be read from the hard disk during a single query is determined by

$$D_{inv} = 4qf^2\frac{I}{v}, \tag{2}$$

where f is the number of visual words per image, I is the number of images in the database and v is the number of

visual words for quantization. The factor 4 is to get the size in Bytes. The variable q is a correction factor to account for an unbalanced inverted file. Some visual words occur more often than others and this will lead to an unbalanced inverted file. Our experiments indicate that $q = 5$ seems to be a realistic estimation of this correction factor.

In addition to the time for reading the data, time for disk seeks (a disk seek entails moving the head and waiting for up to one revolution of the disk) has to be added. For each single query we have to address f different parts of the database. For $f = 1000$ this results in 1000 disk seeks. Typically a disk seek takes about 10ms, amounting to 10s for 1000 seeks. This cost is of special concern for smaller databases. The disk access and read time t_{inv} for one image can be written as

$$t_{inv} = ft_s + 4qf^2\frac{I}{v}t_r, \tag{3}$$

where t_s is the disk seek time (approx. 10ms) and t_r is the read time for 1 Byte (approx. 25ns).

The query time is dominated by the hard drive access and read time (see Table 1 for examples). For databases up to 10 million images the disk seeks are the bottleneck. For larger databases the amount of data to read from disk determines the query time.

4. Multiple independent binnings

To distinguish from the feature vectors of each local image region, we will refer to the vector of visual word occurrences as document vector. The document vectors created by the method of [9] are defined in a 1 million dimensional document space. A set of document vectors (prototypes) defines a partitioning of the document space into Voronoi cells. By randomly picking visual words and assembling document vectors, synthetic prototype document vectors are created. Each set of synthetic prototypes defines a different binning of the document space. The randomly picked prototype sets determine multiple independent binnings of the document space. This is related to locality sensitive hashing in the sense that the prototype sets can be thought of as defining hashing functions. The bins defined by the prototype vectors are now used to organize the image vectors of the image database on the hard drive. Each Voronoi cell of the document space gets assigned a different file on the hard drive. This file will contain all image document vectors that fall into the corresponding Voronoi cell in the feature space. When storing an image vector the distances to all prototype vectors are computed and the image vector is stored in the file associated with the closest prototype vector. This is repeated for each of the different binnings and results in multiple copies of the database on the hard drive, each organized in a different way.

4.1. Querying the bin structure

An image query is performed in two stages. First, the prototype document vector closest to the query is found by nearest neighbor search in the prototype set. The document vectors are normalized using TF-IDF [11] (a standard weighting based on entropy). We will unless specified otherwise assume that L_1 distances are used in the nearest neighbor search. Second, the bin corresponding to the closest prototype document vector is fetched from disk and searched completely. The query process is repeatedly performing these two stages, for each of the different binnings of the database. The closest image document vector over all sets is reported as the matching one.

This scheme is more efficient than the raw inverted file method (section 3.1) as it performs less disk seeks and reads in a smaller amount of feature vectors. For each binning only one bin has to be accessed. A number s of different binnings leads to s disk seeks. When s is smaller than the number f of disk seeks in the former scheme a speed-up can be obtained. In addition only s bins have to be read from disk, where a bin contains only a fraction of the whole database. The number of document vectors in a single bin changes with the number of prototypes b per set. A higher number of prototype document vectors creates smaller bins containing less document vectors to read.

The amount of data (in Bytes) to be read from the hard disk using the binning scheme is given by

$$D_{bin} = 4q_1 sI\frac{f}{b}, \tag{4}$$

where s is the number of different sets, I is the number of images in the database, f is the number of visual words per image, and b is the number of different bins. The variable q_1 again is a correction factor. The images may not be distributed uniformly over all bins. A method to deal with large bins is, to stop reading after some maximum amount. By setting q_1 to 5 we would allow to read in files up to 5 times of their nominal value, but skip the document vectors above this limit.

The disk access and read time t_{bin} for one image can be written as

$$t_{bin} = sq_2 f\frac{bf_{bin}}{v}t_b + st_s + 4q_1 sI\frac{f}{b}t_r, \tag{5}$$

where t_b is the time for one weighted voting operation, t_s is the disk seek time (approx. 10ms), f_{bin} is the maximum number of visual words per prototype document vector and t_r is the read time for 1 Byte (approx. 25ns). Here another correction factor comes in. The variable q_2 accounts for a potential unbalancing of the inverted file used to determine the correct bin for a query. We will set $q_2 = 5$ for the same reason as in section 3.1. In addition to disk seek time and disk read time the binning scheme needs processing time to

determine the bin to read in. The binning time is determined by the number of bins but is almost negligible compared to the disk times. Figure 2 shows the theoretical speed gain for the binning scheme compared to a raw hard disk based inverted file based on Equ. 5 and Equ. 3.

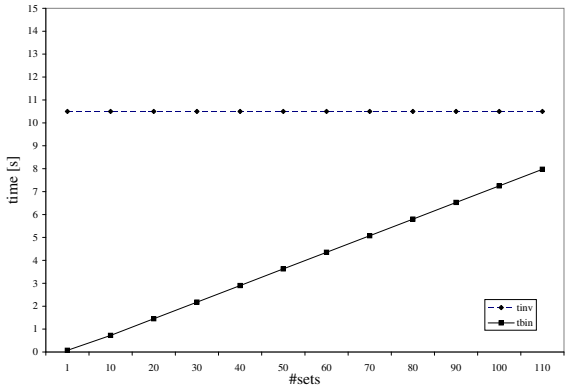

Figure 2. Query time for a 1 million image database predicted for the hard disk based inverted file approach and for the binning scheme with $b = 10000$. The binning scheme will give comparable results at about 20 sets with 1.45 seconds retrieval time.

4.2. Trading retrieval quality for speed

A single binning set will not allow reaching the maximum retrieval quality. Binning errors are expected to occur and a bin might be fetched that does not contain the closest document vector to the query. This is dealt with by the multiple independent binnings. Each processed binning decreases the overall probability of binning errors. This allows trading retrieval quality for speed. One can preset a maximum query time and allow as many binnings as possible to be processed within the alloted time frame. Basically any retrieval time could be met, but eventually the quality will suffer.

5. Implementation of the binning scheme

One detail in the implementation of the binning scheme is the assembly of the synthetic document vectors. Our implementation assembles a synthetic prototype by randomly picking 20 to 50 visual words (VW's) from a uniform distribution. The VW's are picked in a range from 0 to 1 million, which reflects the resolution of the vocabulary tree. The number of VW's between 20 and 50 is also picked randomly. Each selected VW as well as the 4 neighboring VW's are added to the document vector, e.g. if the VW 18225 was selected also the VW's 18223, 18224, 18226 and 18227 are added to the document vector. This creates a boxcar with width 5 around each selected VW's. This procedure creates synthetic document vectors with a maximum of 250 VW's per vector. Another detail is that in the query for the bin we use a discretization by 1000 VW's, and for query the image document vectors we use 1 million VW's. Searching the closest prototype to the query is implemented as scoring using an inverted file as described in [9]. In addition the bins stored on the hard disk are also stored as inverted files. This allows immediate scoring after the bin data has been read in.

6. Determining the necessary number of binnings

Binning errors inevitably occur and prevent us from getting the optimal retrieval quality with a single binning. The binning errors will increase with the number of bins. Figure 3 shows an empirical assessment of the number of correct bin hits depending on the number of bins.

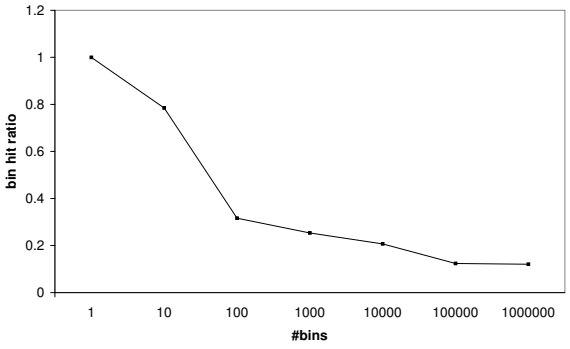

Figure 3. Binning hits: A higher number of bins increases the risk of a bin miss. By using multiple binnings this can be compensated for.

However, if one binning fails on certain document vectors a different set will get them right. Thus in chaining different binnings we will eventually get a correct bin hit in the end. Figure 4 demonstrates this with an experiment on a 10200 image database. Chaining multiple independent binnings will increase the bin hit ratio.

If we assume that the probability p_s of getting the correct bin within one set is independent for each set we can compute the probability for s subsequent binnings by

$$p = 1 - (1 - p_s)^s. \tag{6}$$

The resulting probability p will converge to 1 in the limit. We determine an empirical value for the probability p_s for a certain image dataset and a given b and use this result to predict the graph for different numbers of binnings. Figure 5 shows that the predicted curves follow closely the empirically measured curves from a real experiment. The solid black curves are predicted using Equation (6). The bright curves are the empirical results. Most important is that the empirical curves seem to converge to 1 as well, given a large enough number of sets. The accuracy of our prediction allows us to choose the optimal number of bins for a given

database size. For this experiment on a 2550 image dataset a binning scheme with $b = 1000$ would lead fastest to high quality.

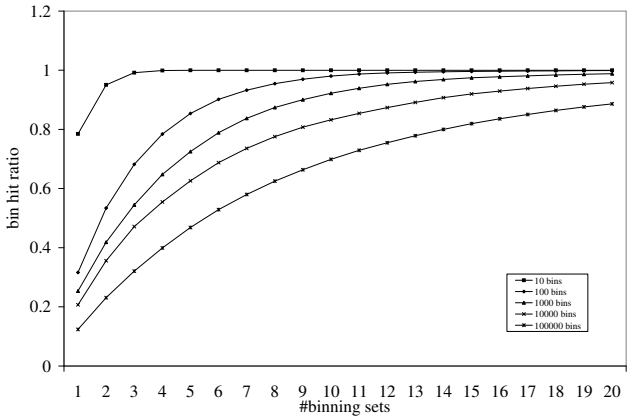

Figure 4. The effect of chaining multiple independent binnings. Each additional set increases the bin hit ratio. The experiment with different numbers of bins shows the tendency of converging to 1.

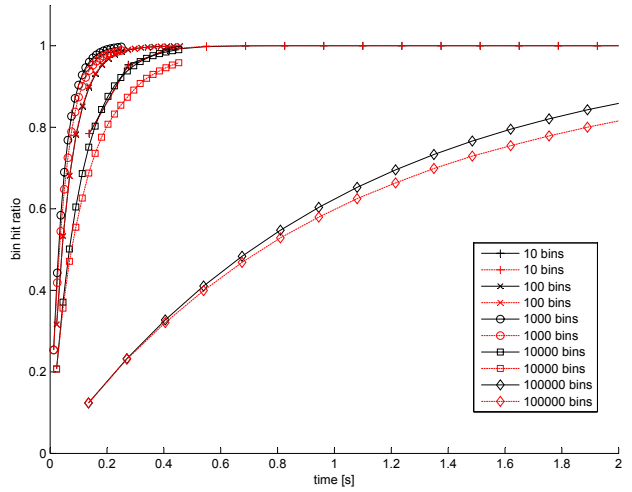

Figure 5. Quality (binning hits) over time plot for the binning scheme on a 2550 image database. Using more binning sets increases the quality but takes more time. The solid black curves are predicted using Equation (6). The bright curves are the empirical validation, which shows that the prediction is quite accurate. The graphs show that the binning scheme with b = 1000 would lead fastest to high quality.

7. Retrieval results

For testing the retrieval quality the 10200 image database from [9][1] was used. The dataset contains 4 images of the same object from different viewpoints and with changing scale. For the first experiment we created a training and a test set from the image set. Each set contains one view of each object. The training set was used to create a binned database on the hard drive. The test set was then used to query the database. The number of bins was set to 1000 for this experiment. An image query returns the n closest images to the query. The computed retrieval quality gives the percentage of correct query results. Figure 7 shows the retrieval quality computed for subsets of different length starting with the 1^{st} image and going to the n^{th} image. The graph also shows the retrieval qualities using different numbers of binnings (1, 2, 5, 10 and 20) and the retrieval quality achieved with the raw inverted file approach without the binning. The quality of the raw inverted file represents the maximum performance that can be achieved by the binning scheme. And it can be seen that the binning scheme reaches this performance already for 20 binnings.

For a second experiment we added 100000 other images to the database. However for these additional images no other views to support a query test were available. Figure 8 shows the retrieval quality for querying the 102550 images database with the test set. The retrieval quality did not drop due to the additional images.

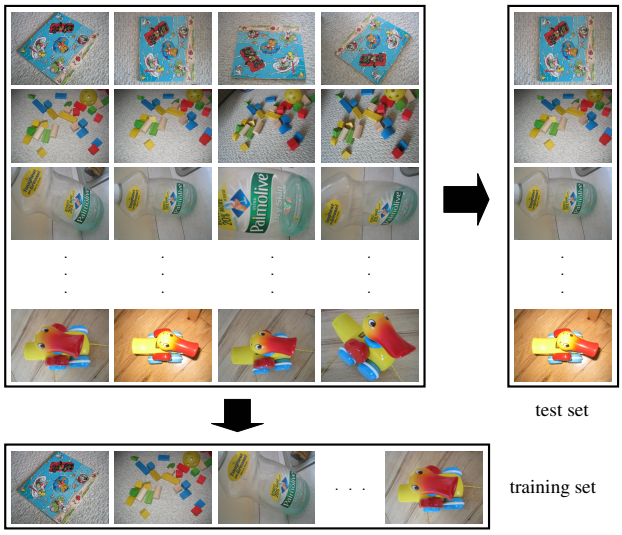

Figure 6. Samples of the 10200 images database set. Each object occurs 4 times, taken from different viewpoints. For a 1-to-1 retrieval experiment we extract a test set and a training set, where each object occurs only once.

In a third experiment we compute the quality measure used in [9]. A database is created containing all 2550 images and the 100000 additional images. We perform queries with the 2550 images of the test set and measure if the top 4 query results contain the 4 images of the same object. The maximum value for this measure is 4. The experiment was also carried out for different numbers of binnings. Figure 9 shows the results.

[1] available at http://vis.uky.edu/ stewe/ukbench

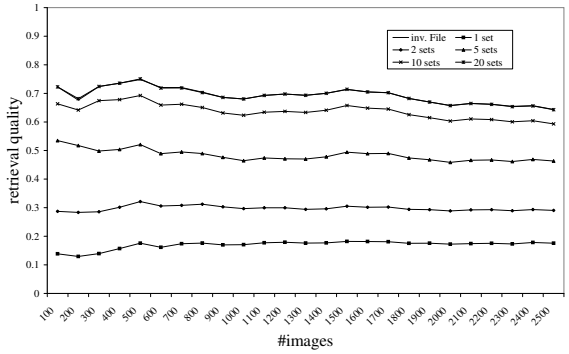

Figure 7. Retrieval quality (correct rank one queries) with b = 1000. The database contained 2550 images. The 2550 images from the test set were used for querying. With 20 different binnings there is almost no difference to the inverted file method.

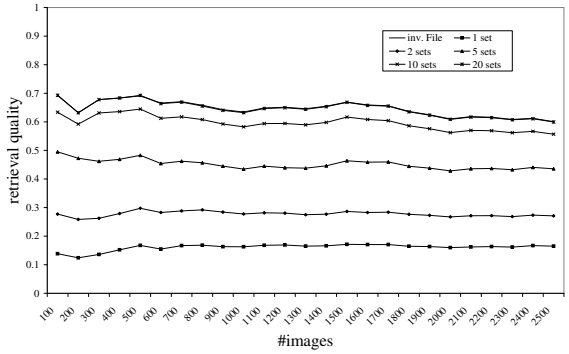

Figure 8. Retrieval quality (correct rank one queries) with b = 1000. The database contained 102550 images. The 2550 images from the test set were used for querying. With 20 different binnings there is almost no difference to the raw inverted file method.

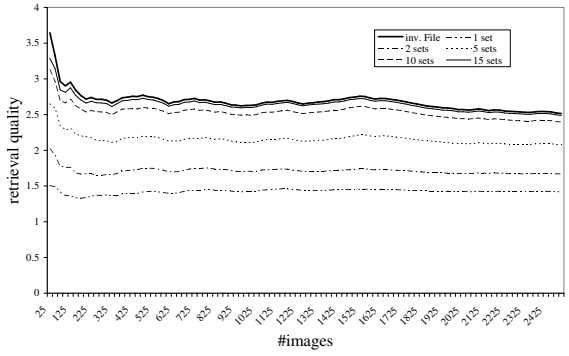

Figure 9. Retrieval quality (number of correct images in the top 4 results) with b = 1000. The database contained 110200 images. The 2550 images from the ground truth test set were used for querying. With 15 different binnings there is almost no difference to the raw inverted file method.

8. Conclusion

An image search scheme based on splitting the database into multiple independent binnings has been presented. The scheme significantly reduces the amount of data to process per image query. When the database is stored on a hard drive and the data access is slow the presented scheme provides a significant speed up. The experiments showed that the binning scheme can provide the same retrieval quality as a raw search. In addition it is possible to trade retrieval quality for speed, which essentially allows meeting any retrieval time. It is tempting to imagine that this binning scheme will allow image search on databases beyond the sizes so far attempted, without the use of massive computer farming. Maybe even content based image search on web scale could be within reach.

References

[1] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proc. ACM Symp. on Computational Geometry*, pages 253–262, 2004.

[2] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *Proc. 10th IEEE International Conference on Computer Vision, Beijing, China*, pages 1458–1465, 2005.

[3] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *30th Ann. ACM Symp. on Theory of Computing*, 1998.

[4] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition, San Diego, California*, pages 775–781, 2005.

[5] D. Lowe. Object recognition from local scale-invariant features. In *Proc. 7th International Conference on Computer Vision, Kerkyra, Greece*, pages 1150–1157, 1999.

[6] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[7] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proc. 13th British Machine Vision Conference, Cardiff, UK*, pages 384–393, 2002.

[8] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1-2):43–72, 2005.

[9] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition, New York City, New York*, 2006.

[10] S. Obdrzalek and J. Matas. Sub-linear indexing for large scale object recognition. In *Proc. 15th British Machine Vision Conference, Oxford, UK*, pages 1–10, 2005.

[11] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. 9th IEEE International Conference on Computer Vision, Nice, France*, pages 1470–1477, 2003.