

# Pruning SIFT for Scalable Near-Duplicate Image Matching

Jun Jie Foo

Ranjan Sinha

School of Computer Science & IT  
RMIT University, Melbourne, Australia, 3001  
Email: {jufoo,rsinha}@cs.rmit.edu.au

## Abstract

The detection of image versions from large image collections is a formidable task as two images are rarely identical. Geometric variations such as cropping, rotation, and slight photometric alteration are unsuitable for content-based retrieval techniques, whereas digital watermarking techniques have limited application for practical retrieval. Recently, the application of Scale Invariant Feature Transform (SIFT) interest points to this domain have shown high effectiveness, but scalability remains a problem due to the large number of features generated for each image. In this work, we show that for this application domain, the SIFT interest points can be dramatically pruned to effect large reductions in both memory requirements and query run-time, with almost negligible loss in effectiveness. We demonstrate that, unlike using the original SIFT features, the pruned features scales better for collections containing hundreds of thousands of images.

*Keywords:* near-duplicate image matching, near-replicate image retrieval

## 1 Introduction

On the web, we often find copies, versions, or even fragments of images that are scaled-down thumbnails, or variants of the same digital image that are, legally or otherwise, kept by different sources. These are often images that are almost identical but not recognised as such due to common image manipulations such as conversion to greyscale, change in color balance and contrast, rescaling, rotating, cropping, and filtering. Such image variants are commonly known as near-duplicate images [Jaimes, Chang & Loui 2002, Ke, Sukthankar & Huston 2004, Zhang & Chang 2004].

In most cases, the storage and retrieval of duplicate and near-duplicate images may be unnecessary and, in the context of collections derived from the web, may further present infringements of copyright. However, the detection of these near-duplicate images is a formidable task as two image versions are rarely identical. They may differ in filename, format, and size; simply saving an image may lead to bitwise differences due to the variations in the coding standards in different software.

While the detection of copied digital images have been extensively researched in the field of digital watermarking [Hartung & Kutter 1999, Johnson, Duric

& Jajodia 1999, Kang, Huang & Shi 2002, Kang, Huang, Shi & Lin 2003], such methods are ill-suited for retrieval applications due to practicality issues [Lu & Hsu 2005, Qamra, Meng & Chang 2005]. Similarly, content-based retrieval techniques [Smeulders, Worring, Santini, Gupta & Jain 2000] — a well researched area — are unsuitable as they are designed for a much broader class of image matches with similar traits of colour, texture, or shapes; these techniques have been shown to have limited effectiveness for this task [Chang, Wang & Wiederhold 1998, Sebe, Lew & Huijsmans 1999, Luo & Nascimento 2003].

For the task of retrieval of near-duplicate images, Ke et al. [2004] have demonstrated near-perfect accuracy using PCA-SIFT local descriptors [Ke & Sukthankar 2004] on a moderate-sized image collection of about 20,000 images, however, scalability and efficiency aspects were not discussed. Qamra et al. [2005] propose perceptual distance functions for near-duplicate retrieval using color and texture image features on large proprietary image collections; however, limited effectiveness is observed, and efficiency aspects were only briefly discussed. Lu & Hsu [2005] demonstrated an image hashing technique for the retrieval of near-duplicate images on a collection of 20,000 images with limited effectiveness, wherein the number of alterations, though large, are limited in severity.

Thus far, in this domain, only Ke et al. [2004] have demonstrated a method that is indeed highly accurate for even relatively severe alterations. However, an inherent problem with applying SIFT [Lowe 2004] features (represented with PCA-SIFT local descriptors) is the large number of features — ranging from hundreds to thousands — it generates per image, whereby each feature is represented by a high dimensional feature vector [Ke & Sukthankar 2004]. Though these large set of image features are indexed using Locality-Sensitive Hashing [Gionis, Indyk & Motwani 1999, Ke et al. 2004], we show that such large numbers of features are impractical for moderate to large image collections.

The SIFT detector was originally designed for robust matching of even small occluded objects; hence, the quantity of features is crucial for such applications [Lowe 2004, Ke & Sukthankar 2004]. We hypothesize that only a small subset of features are required for near-duplicate image matching, as most images of interest in this domain contain high perceptual similarity to their originals [Qamra et al. 2005]. In this work, we propose a pruning strategy that reduces the number of SIFT features (consequently PCA-SIFT local descriptors), thus enabling these descriptors to scale well for a large collection containing hundreds of thousands of images. Such a pruning strategy simultaneously reduces the index size and query response time to 1/10 and 1/50, respectively, of the original approach, with little impact on effec-

tiveness.

In the following section, we describe the distinctive local descriptors (mainly SIFT and PCA-SIFT). We discuss the Locality Sensitive Hashing index structure used in this work in Section 3. In Section 4, we introduce our approach of pruning the SIFT interest points and in Section 5, we describe the evaluation methodology and experimental setup. The results are presented and discussed in Section 6, followed by our conclusions in Section 7.

## 2 Distinctive Local Descriptors

Local descriptors computed for images have been demonstrated to be useful for object recognition [Lowe 2004], and robust image matching [Ke et al. 2004, Grauman & Darrell 2005]. Given an image, the idea is to detect image regions (centered around interest points) that possess properties invariant to geometric variation and photometric changes, so that distinctive local descriptors can be computed for each region. Popular region detectors include the Harris-Point, Harris-Laplace, Harris-Affine, Hessian-Laplace, and Hessian-Affine; popular descriptors include the SIFT (Scale Invariant Feature Transform) [Lowe 2004], and PCA-SIFT [Ke & Sukthankar 2004], to name a few. For a complete survey on region detectors and local descriptors, the reader is directed to the work of Mikolajczyk & Schmid [2003]

In this work, we use the SIFT (scale invariant feature transform) detector — which uses regions similar to Hessian-Laplace — as it has been demonstrated to outperform most existing detectors [Mikolajczyk & Schmid 2003]. We apply the PCA-SIFT descriptors on the SIFT interest points, instead of the original SIFT descriptors, as it has been reported to be both highly distinctive [Ke & Sukthankar 2004] and highly effective for near-duplicate image detection [Ke et al. 2004].

### SIFT and PCA-SIFT descriptors

The Scale Invariant Feature Transform (SIFT) [Lowe 2004] devised for robust image feature detection is invariant to scale, rotation, and affine transforms. There are four major computational stages in SIFT for extracting a set of image features, namely the scale-space extrema detection, *keypoint* localization, orientation assignment, and generation of local descriptor.

In the first phase of the SIFT detector, the difference-of-Gaussian (DoG) function uses a Gaussian pyramid to identify any local peaks (keypoints) in various locations and scales. (This is achieved by finding the local scale-space extrema of the DoG). In the second phase, poorly localized and unstable keypoints below several threshold levels are rejected; mainly based on contrast level and ratio of principal curvature [Lowe 2004]. After all stable keypoints are identified, each keypoint is assigned a dominant orientation for rotation invariance in the third phase. Additional keypoints are generated if there are multiple orientations within 80% threshold of the dominant orientation; thus, there can be multiple keypoints with identical scale, location, but different orientation.

Instead of computing SIFT local descriptors, we use the PCA-SIFT descriptors which have been shown to be the most distinctive [Mikolajczyk & Schmid 2003] compared to other descriptors. To generate local descriptors, PCA-SIFT uses the same information as the original SIFT descriptor, that is, location, scale, and dominant orientations. PCA-SIFT concatenates the horizontal and vertical gradient maps

for the  $41 \times 41$  region — centered around the keypoint, rotated to align its orientation to a canonical direction — to produce a  $2 \times 39 \times 39 = 3042$  element local descriptor (feature vector).

In cases where there are multiple dominant orientations, a separate vector is calculated for each. Each vector is then projected — using principal component analysis, a common technique for dimensionality reduction — to a low-dimensional feature space using a pre-computed eigenspace<sup>1</sup>. Ke et al. [2004] have empirically determined that  $n = 36$  feature spaces for the local descriptor performs well for near-duplicate image retrieval; wherein any two PCA-SIFT local descriptors are deemed similar (a match) within an Euclidean distance ( $L_2$ -norm) of 3000. Hence, in this work, we use the same settings.

The number of PCA-SIFT local descriptors are dictated by the keypoints detected by SIFT, typically ranging from hundreds to thousands per image (depending on image complexity). The indexing of such a substantial number of PCA-SIFT local descriptors for large image collection is impractical and costly. We later show that the number of keypoints that SIFT generates can be significantly reduced by varying the threshold value in the second stage, resulting in great gains in efficiency with only slight loss in accuracy for matching near-duplicate images.

## 3 Indexing Local Descriptors

To index a set of 36-dimensional PCA-SIFT descriptors, we use the Locality Sensitive Hashing (LSH) index scheme [Indyk & Motwani 1998, Gionis et al. 1999, Datar, Immorlica, Indyk & Mirrokni 2004, Ke et al. 2004, Bawa, Condie & Ganesan 2005] for approximate nearest-neighbour matching in high-dimensional spaces.

Given a set of points (PCA-SIFT descriptors)  $P$ , the distance between two points in the approximate nearest neighbor search can be defined as:

$$d(q, p) \leq (1 + r)d(q, P)$$

where  $q$  is the query point, and  $d(q, P)$  is the distance of  $q$  to the closest point in  $P$ . The key idea of LSH is to use a family of hash functions to ensure that the probability of collision of two points is closely related to the distance between them.

A hash function can be defined as  $g_i(p) = (h_1(p), \dots, (h_k(p)))$ , for  $i = 1, \dots, l$ , where  $k$  determines the probability of collision, and  $l$  determines the fraction of false negatives [Indyk & Motwani 1998]. More specifically, this family of hash functions is called  $(r_1, r_2, p_1, p_2)$ -sensitive [Gionis et al. 1999] for the distance between the elements in  $P$  for any  $q, p \in P$ :

- if  $p \in \beta(q, r_1)$  then  $Pr_H[h(q) = h(p)] \leq p_1$
- if  $p \in \beta(q, r_2)$  then  $Pr_H[h(q) = h(p)] \geq p_2$

where  $\beta(q, r)$  denotes the set of elements within the distance of  $r$  to  $q$ . Additionally, the requirements of  $p_1 > p_2$  and  $r_1 < r_2$  has to be satisfied for locality-sensitivity.

The family of functions can be efficiently computed with a Hamming space  $H^d$  for  $d$  dimensions [Gionis et al. 1999], whereby each  $d$ -dimensional vector  $p(x_1, \dots, x_d)$  can be mapped to a Hamming cube  $H^{d'}$  with  $d' = Cd$  (where  $C$  denotes the largest coordinate in  $P$ ), transforming vector  $p$  to a binary Hamming string  $p'$ .

<sup>1</sup>The eigenspace used in this work is provided by Ke & Sukthankar [2004].

Given a transformed vector of  $p'$ , a hash  $g_i(p)$ , for  $i = 1, \dots, l$ , can be obtained by a projection of vector  $p'$  onto the coordinate set  $I_i$  (where  $I$  consists of  $k$  elements that are sampled randomly with replacement from  $\{1, \dots, d'\}$ ) essentially hashing point  $p$  to bucket  $g_i(p)$ . As the number of buckets can be high depending on the cardinality of set  $P$ , a second level of standard hashing is used to map the contents of  $g_i(p)$  to a hash table [Gionis et al. 1999]. Hence, there are a total of  $l$  LSH tables, each using the LSH functions from the  $H^{d'}$  family. The size of each hash table  $M$  is determined using:

$$M = \frac{n}{\alpha B}$$

where  $n$  is the total number of points in a collection, and  $B$ ,  $l$ , and  $k$  (critical for efficacy), are empirically determined to be of size 20, 20, and 450, respectively [Ke et al. 2004]; the utilization parameter  $\alpha$  is selected to be 0.5.

All points sharing identical hash values (collisions) within a given hash table are estimated, by the Manhattan distance ( $L_1$ -norm embedded in the Hamming space), to be closer to each other than those that do not. Thus, the search space of an approximate nearest-neighbor match is greatly reduced to those that share identical hash values. To further eliminate the number of false positive matches, an additional verification step is required as the LSH index structure returns only approximate matches based on  $L_1$  whereas the PCA-SIFT features require two vectors to be within an  $L_2$  (Euclidean) norm of 3000 to be deemed a match [Ke & Sukthankar 2004]; hence, all keypoint matches are post-processed to discard false positive matches. A final filtering phase is applied using robust estimators such as RANdom SAMple Consensus (RANSAC) [Fischler & Bolles 1981] to geometrically verify that two images are indeed near-duplicates to ensure high precision in the returned answers. We apply the same filtering in our work.

For query evaluation, all candidate matches are returned by the LSH index for every query keypoint. Hence, each query image is treated as a *bag of points*, simulating a multi-point query evaluation. To do this efficiently, we use an identical framework as Ke et al. [2004], in that, we maintain two auxiliary index structures — File Table (FT) and Keypoint Table (KT) — to map SIFT keypoints (and PCA-SIFT descriptors) to their corresponding images; an entry in KT consists of the file ID (index location of FT) and keypoint information ( $x$  and  $y$  location, scale, orientation, and the PCA-SIFT local descriptor). The cost of query evaluation of a single image depends highly on the cardinality of the set of keypoints  $P$  in any given image. We observe an average of 1900 keypoints for a large crawled image collection, which implies 1900 point queries to the LSH index for the evaluation of a single image query; this is computationally intensive and highly impractical for online interactive querying on large image collections.

In the next section, we introduce our approach for pruning the SIFT interest points.

#### 4 Reducing SIFT Interest Points

With the SIFT detector, the number of computed keypoints are typically in the order of  $10^3$  (this depends on the image content, size, and complexity). Such quantities of keypoints is often crucial for object-recognition to enable even small occluded objects to be reliably matched [Lowe 2004]. However, for retrieval tasks on large collections, such a large number of keypoints can reduce any efficient index structure to a sequential search.

We hypothesise that the application domain of near-duplicate image detection may not require the full set of keypoints, and often a small subset will suffice. The rationale being that the application of near-duplicate image matching is often targeted at images displaying high perceptual similarity [Qamra et al. 2005]; even cropped images (to a certain extent) will possess many similar traits (objects, subjects, texture, and shape) to the original image.

To reduce the number of SIFT interest keypoints, we vary the threshold applied to discard candidate local peaks, specifically, the low contrast (intensity) threshold. Earlier, a threshold value of 0.03 was empirically observed to work [Lowe 2004] in the domain of object recognition, but it typically yields a large number of keypoints. However, using geometric verification techniques such as RANSAC [Fischler & Bolles 1981], only a small number of keypoints — 3 to 5 — are required for reliable image matching [Lowe 2004, Ke et al. 2004].

There are two ways that this inclusion threshold value can be used to reduce the cardinality of the keypoint set:

1. Select top  $N$  most significant keypoints ranked by the contrast value.
2. Raise the inclusion threshold value to discard keypoints.

Both methods can be used to reduce the number of keypoints. The second method, however, yields an unstable number of keypoints as images with lower average intensity level may not have any detected keypoints. Thus, varying the inclusion threshold could yield undesirable results since no assumption can be made regarding intensity levels of any given image. Hence, we apply the former approach, which sets an “upper bound” on the number of keypoints selected in this phase. Images that do not have  $N$  detected keypoints are not pruned; we experiment with varying the number of  $N$  significant keypoints to determine the optimal value. Note that the term upper bound is used loosely, since some keypoints may share the same location and scale information with multiple orientation, resulting in approximately 15% additional keypoints generated in the subsequent phase [Lowe 2004].

This pruning strategy enables images to be indexed using only a small subset of keypoints (and PCA-SIFT local descriptors) and as observed with only slight loss in effectiveness during retrieval.

#### 5 Evaluation Methodology

We now describe the series of experiments used to empirically demonstrate the effectiveness of our approach. First, we evaluate the effectiveness of keypoint reduction by varying the number of detected keypoints between 1900 (original number of keypoints as observed in our collection) and using a subset of 1000, 100 and 10 most significant keypoints. These values are henceforth referred to as *threshold value*.

Second, we report the percentage of keypoint matches between a query image and each of its image alterations; this percentage is relative to the keypoints in the query image. Two keypoints are deemed to be a match if the nearest-neighbors are within the  $L_2$  norm of 3000. For an accurate evaluation, we use the sequential scan for the nearest-neighbor search on the collection of keypoints as the LSH index is an approximate nearest-neighbor algorithm. Due to the exhaustive computation involved with using several keypoint thresholds, we use 100 random query sets

and evaluate the keypoint matches for each of its alterations (see below). All subsequent experiments are performed using the LSH index with 200 queries.

Third, we evaluate the effects of keypoint reduction at all threshold levels, 1000, 100, and 10 and compare them against the original approach in terms of retrieval effectiveness (wherein the original image is used to retrieve all altered versions). We apply the standard recall and precision metrics, which are defined as:

$$recall = \frac{\text{relevant images retrieved}}{\text{total relevant images in collection}}$$

$$precision = \frac{\text{relevant images retrieved}}{\text{total images retrieved}}$$

We also measure query run-time<sup>2</sup>, and index size.

Fourth, we evaluate the effectiveness of our approach — against the original approach — using each altered image as a query to retrieve only its respective original image from which it is derived; for this purpose specifically, we do not consider other altered images that are relevant to a given query. In addition, using a more stringent evaluation, we assess the retrieval effectiveness of our approach using each altered image as a query to retrieve all other altered images that are derived from the same original image; this is also compared against the original approach.

Finally, we vary the threshold levels on both the query images and the indexed test images (images relevant to the query that are within the collection). For instance, we use a query image with a threshold value of 1000 to retrieve images that are indexed using a threshold value of 100 to study their effects on retrieval accuracy and query run-time. An identical framework to Ke et al. [2004] is used; the only difference is the amount of PCA-SIFT features used.

All experiments are run on a two-processor Xeon 3 GHz machine with 4 GB of main memory running Linux 2.4.

## Image Collections

To create our test collection, we select 200 images at random from the crawled SPIRIT collection [Joho & Sanderson 2004] and perform 50 common alterations. We use the original image as a query whereby all altered images are considered relevant answers. Hence, each of the selected 200 images produce 50 altered versions, yielding 10,000 images. We then select 10,000 and 90,000 additional images from the SPIRIT collection to serve as noise, thereby creating two image collections —  $C_1$  and  $C_2$  — with aggregated sizes of 20,000 and 100,000 images, respectively. The list of alterations — similar to those of Ke et al. [2004] and Qamra et al. [2005] — is as follows:

1. **format change**: format change from .jpg to .gif (1)
2. **colorise**: each of the red, green, and blue channels are tinted by 10% (3)
3. **contrast**: increase and decrease contrast (2)
4. **severe contrast**: increase and decrease contrast 3× of original image (2)
5. **crop**: crop 95%, 90%, 80%, and 70% of image, preserve center region (4)
6. **severe crop**: crop 60%, 50%, and 10% of image, preserve center region (3)
7. **despeckle**: apply “despeckle” operation of ImageMagick (1)

<sup>2</sup>Note that query run-time does not include feature extraction time for simplicity.

8. **frame**: a frame size 10% of image is added using random colors (4)
9. **rotate**: rotate image (by 90°, 180°, and 270°) about its center (3)
10. **scale-up**: increase scale by 2×, 4×, and 8× (3)
11. **scale-down**: decrease scale by 2×, 4×, and 8× (3)
12. **saturation**: alter saturation by 70%, 80%, 90%, 110%, and 120% (5)
13. **intensity**: alter intensity level by 80%, 90%, 110%, 120% (4)
14. **severe intensity**: alter intensity level by 50% and 150% (2)
15. **rotate+crop**: rotate image (by 90°, 180°, and 270°), crop 50% in center region (3)
16. **rotate+scale**: rotate image (by 90°, 180°, and 270°), decrease scale 4x (3)
17. **shear**: apply affine warp on both x and y axes using 5°, and 15° (4)

The number in parentheses is the number of instances for each alteration type.<sup>3</sup>

For both collections  $C_1$  and  $C_2$ , all images are converted into greyscale<sup>4</sup> and resized to 512 pixels in the longer edge prior to feature extraction and indexing.

## 6 Results

We now present our results on the retrieval effectiveness of our keypoint pruning strategy on several image alterations and discuss the efficiency of this approach in terms of query run-time, and index size.

### Retrieval Effectiveness

As shown in Table 1, the effects of keypoint reduction is presented for all threshold values on all 50 alterations; the percentages are averaged over 100 queries. Columns 1, 6, and 11 refer to the different alterations (indicated by *Alt*); the adjacent columns denote the percentage of keypoint matches within the  $L_2$  norm of 3000. Columns 2, 7, and 12 (indicated by Default) indicate the original number of keypoints per image, which is observed to be an average of 1900 for our collection.

We experiment with threshold values of 1000, 100, and 10 (reflected in subsequent columns), reducing the average keypoints per image to 1059, 130, and 14, respectively. Table 1 shows that keypoint reduction on all threshold values have little effect on the percentage of keypoints matched within the threshold criterion for most image alterations. This implies that our pruning strategy is appropriate for near-duplicate image matching.

The variation between the percentage of matching keypoints of image alterations are expected as some alterations severely affect the local descriptors, yielding lower overall keypoint matches. These trends are relatively stable across all levels of reduction, which leads us to believe that a small subset of keypoints is sufficient for this application. This is an important finding as the same criterion of  $L_2$ -norm within 3000 is the basis by which the LSH index approximates matching keypoints.

For some alterations, the slight increase in the percentage of matching keypoints, for some threshold values as compared to the default, is explained by the fact that matched keypoints are relative to the number of detected keypoints in the image.

Table 1: Percentage (%) of keypoint matches within  $L_2$  norm threshold at every level of reduction. Columns 1, 6, and 11 indicate the different alterations (Alt). Keypoint thresholds of 1000, 100, and 10 are used. Default indicates the original number of keypoints per image (average of 1,900).

Alt	Default	1000	100	10	Alt	Default	1000	100	10	Alt	Default	1000	100	10
1	84.4	87.2	90.2	88.9	18	46.3	50.5	56.6	55.2	35	20.9	21.3	23.2	19.3
2	78.2	81.3	85.1	83.8	19	22.7	23.8	31.7	32.5	36	3.2	2.5	0.3	0.6
3	81.0	84.1	87.1	85.7	20	7.8	7.9	12.2	14.3	37	52.3	52.7	61.0	62.4
4	68.5	72.8	76.5	73.5	21	4.6	3.7	3.6	5.0	38	47.3	54.2	46.4	33.9
5	67.9	69.9	73.6	71.1	22	59.3	62.5	66.1	58.6	39	42.6	46.0	47.4	41.6
6	40.5	45.7	49.6	42.7	23	60.2	62.8	65.3	58.6	40	37.1	34.9	32.0	29.4
7	36.3	42.1	47.0	38.2	24	59.2	62.0	64.5	56.8	41	18.0	17.1	19.2	16.4
8	31.6	37.3	42.8	35.2	25	80.8	84.0	87.9	86.4	42	20.4	17.9	17.6	14.7
9	26.4	31.0	36.4	32.3	26	82.4	85.5	88.8	86.7	43	17.7	16.7	18.7	15.7
10	59.0	59.8	71.4	70.4	27	84.3	87.1	90.9	89.5	44	18.1	16.6	19.7	18.1
11	73.2	78.5	84.5	84.4	28	83.8	86.8	90.2	90.4	45	18.3	16.7	17.9	15.3
12	38.8	37.0	42.7	38.2	29	81.9	85.0	88.4	87.6	46	17.5	16.1	19.3	18.5
13	32.7	31.6	36.2	30.4	30	68.5	70.5	76.6	75.5	47	43.9	47.3	53.1	41.2
14	31.8	30.5	33.3	14.7	31	74.6	77.5	81.8	80.4	48	12.1	12.1	9.0	6.9
15	36.1	35.0	41.6	38.3	32	74.6	79.0	82.7	80.2	49	35.7	35.4	34.8	15.1
16	46.6	51.4	57.6	53.7	33	65.8	72.7	73.8	65.5	50	19.4	17.4	9.5	5.3
17	49.1	53.2	54.6	51.5	34	22.3	24.9	29.6	25.6	-	-	-	-	-

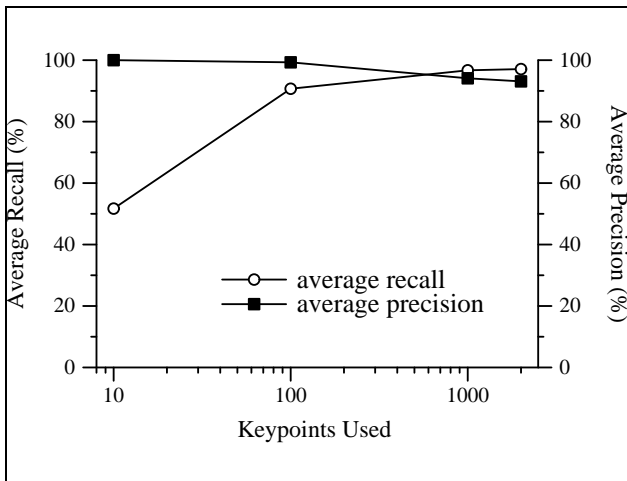


Figure 1: Average recall and precision (%) of retrieved answers using original images as queries on collection  $C_1$  using 1900 (default), 1000, 100, and 10 keypoints; all values are averaged over 200 queries.

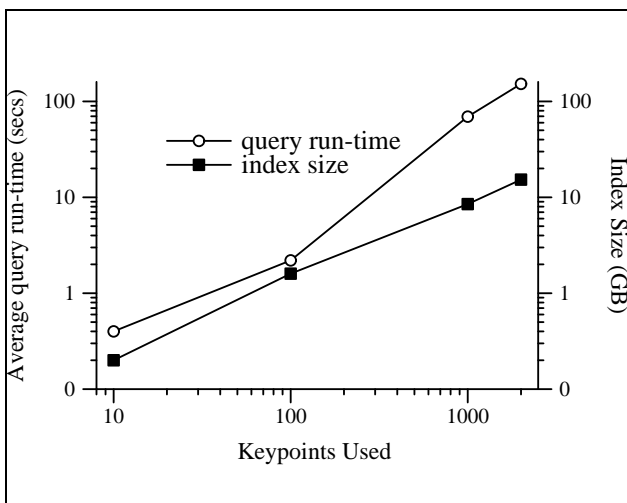


Figure 2: Average query run-time (secs) and Index sizes (GB) using 1900 (default), 1000, 100, and 10 keypoints on Collection  $C_1$ ; all timings are averaged over 200 queries.

Figure 1 shows the average recall and precision of 200 queries on image collection  $C_1$ . The original image is used as a query to retrieve *all* its alterations; we apply the same variation of threshold values (1000, 100, and 10). Using a small threshold value of 100, we observe a high precision of 99.3% with only a slight drop in average recall of 90.7%. This implies that using roughly 100 keypoints, we retrieve 45 instead of 48 (default gives an average recall of 97.1%) on average. The average precision values for this threshold value also indicate that using less than 10% of the default number of keypoints, most of the relevant answers are retrieved with near-perfect accuracy. However, a threshold value of 10 results in a low average recall of 51.7%, indicating that too severe a reduction, results in considerable loss in effectiveness as almost 45% of the relevant answers are not retrieved.

### Retrieval Efficiency

In Figure 2, we show the effect of varying threshold values on query evaluation time and memory requirements. With default settings, a single query evaluation on collection  $C_1$  takes 152.4 seconds on average (over 200 queries). Using keypoint reduction with thresholds of 1000, 100 and 10, we observe timings of 69.4, 2.2, and 0.4 on average for a single query. This is a significant result, given that a threshold value of 100 also yields high effectiveness (average recall and precision of 90.7 and 99.3, respectively), with only a fraction of the speed of the original approach.

The on-disk memory requirements for the index (index size) of the same collection  $C_1$  is also considerably reduced from 15.3 GB (default) to 8.5, 1.6, and 0.2 GB, for threshold values of 1000, 100, and 10, respectively. Note that the index size includes the Keypoint Table (KT) — wherein each entry is 92 bytes in length — and the LSH index; we do not include the File Table (FT) as this remains unchanged.

We do not stress on the timing patterns that are observed in our experiments as the existing implementation of the LSH index is not optimised in-memory; the timings were merely an indication of savings by the current framework. We believe an investigation

<sup>3</sup>All alterations are created using ImageMagick, <http://www.imagemagick.com>.

<sup>4</sup>The PCA-SIFT features are extracted from greyscale images.

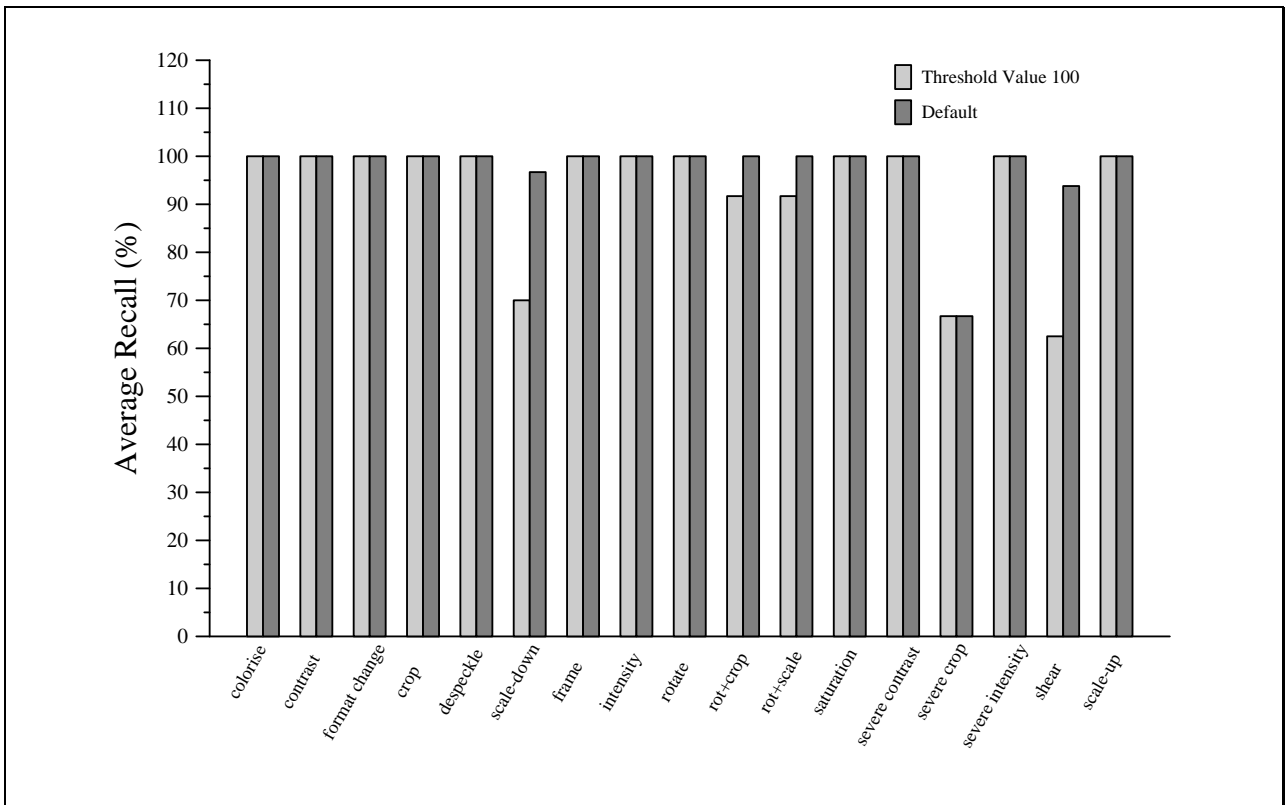


Figure 3: Average recall of each alteration over 20 queries — on Collection  $C_1$  — to retrieve their respective original images using the original approach and threshold value of 100; each of the 17 alteration groups are described in Section 5, wherein there are 50 individual alterations in total.

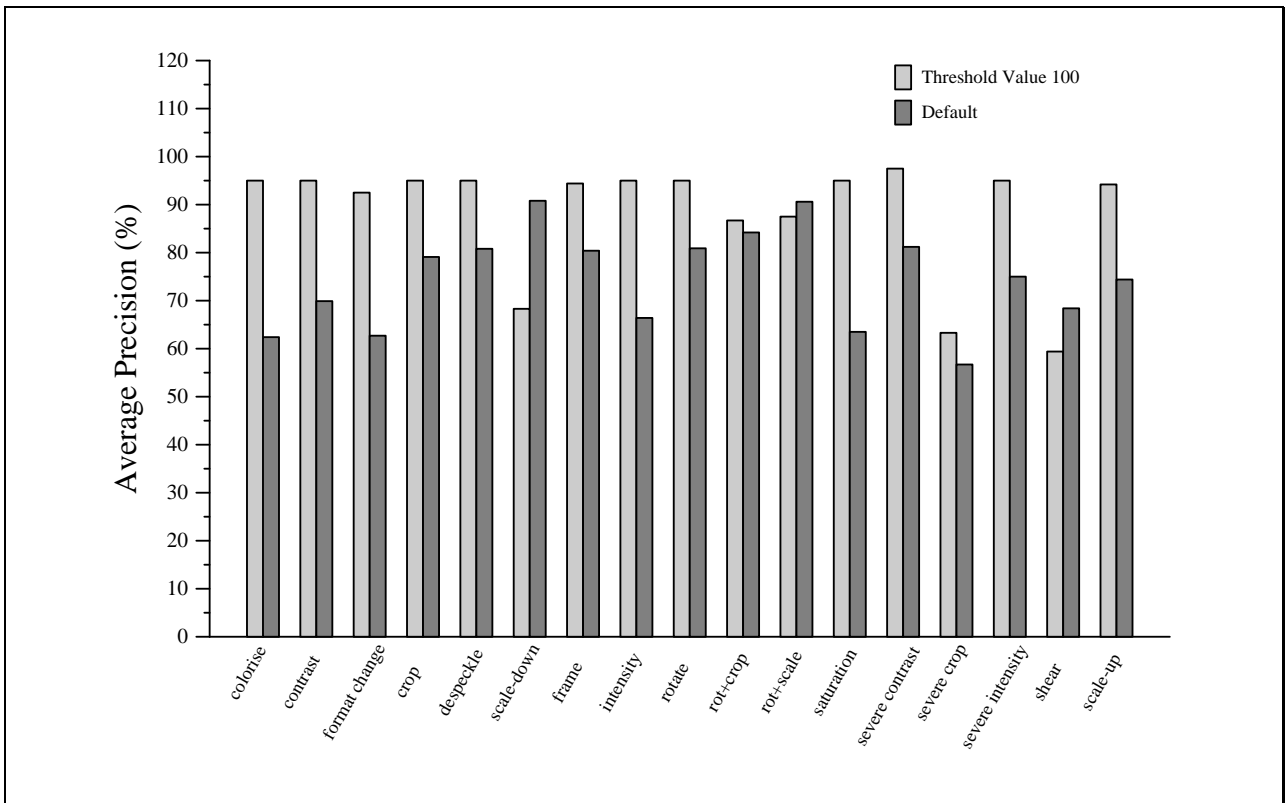


Figure 4: Average precision of each alteration over 20 queries — on Collection  $C_1$  — to retrieve their respective original images using the original approach and threshold value of 100; each of the 17 alteration groups are described in Section 5, wherein there are 50 individual alterations in total.

Table 2: Average recall and precision (%) of 200 queries on Collection  $C_1$  using four threshold values of 1000, 100, and 10; default indicates the original number of keypoints (1900 on average). Column 2 indicates the threshold value used for the query images.

Index KP	Query KP	Avg. Recall	Avg. Prec.	Avg. run-time	Candidate Set
Default	Default	97.1	93.1	152.4	62,549
	1000	96.7	93.9	102.1	31,890
	100	91.4	99.0	15.6	3,706
	10	56.1	100.0	1.0	395
1000	Default	96.9	93.4	91.5	41,772
	1000	96.7	94.1	69.4	27,634
	100	91.4	98.9	10.8	3,413
	10	56.2	100.0	0.7	372
100	Default	94.5	98.4	4.7	7,836
	1000	94.5	98.5	2.8	5,279
	100	90.7	99.3	2.2	2,647
	10	56.2	100.0	0.5	321
10	Default	68.4	100.0	0.9	920
	1000	68.4	100.0	0.7	634
	100	67.6	100.0	0.5	385
	10	51.7	100.0	0.4	265

on efficient in-memory data structures can further improve scalability on even larger image collections.

The threshold value of 100 provides the best trade-off between effectiveness and efficiency and hence we apply this threshold value in further experiments.

## Further Experiments

To investigate the effects of keypoint reduction on each image alteration, we use each altered version to retrieve its respective original image. We experiment with a threshold value of 100 and compare it to the original approach. Due to the large number of alterations, it is impractical to query using all 200 queries for each alteration; instead, we use 20 different queries for each alteration. For this experiment specifically, only the original images are indexed along with the noise collection of  $C_1$ ; altered images are hence replaced by crawled images from the SPIRIT collection. To avoid confusion, we refer to this collection as  $C'_1$ .

Figures 3 and 4 depict the retrieval effectiveness — average recall and precision, respectively — of each alteration on Collection  $C'_1$ . The 50 alterations are categorised into 17 groups as listed in Section 5. We observe a pleasingly high — relative to the original approach — average recall and precision across all alteration groups. For most alteration groups, the average recall is lossless, indicating that the original image is found with the altered query image using both threshold value of 100 and the original approach. An even more surprising result as shown in Figure 4 is that the same threshold value achieves an overall higher average precision than the original approach across almost all alterations, except for **scale-down**, **rotate+scale**, and **shear**. This indicates that our pruning strategy does not adversely affect accuracy but improves it; this can be explained the fact that our pruning strategy reduces the number of keypoints that are examined considerably, thereby also eliminating the number of potential false matches.

Figures 5 and 6 show the effectiveness — average recall and precision — of each alteration as queries in the retrieval of all other alterations on Collection  $C_1$ .

We observe a lower average recall for a threshold value of 100 as compared to the original approach; however, note that this is a rather stringent evaluation measure given that a miss in one image alteration will be reflected in two groups. For instance, if a query image of alteration  $A$  does not retrieve an image of alteration  $B$ , a query the other way around would similarly fail. Nevertheless, for most alterations the discrepancies between them are relatively uniform, the only exceptions are **scale-down**, **severe contrast**, and **shear**, where we observe greater discrepancies. In contrast, Figure 6 shows a higher precision for all alterations using the threshold value of 100, indicating that the keypoint reduction slightly decreases completeness with no effect on the retrieval accuracy.

To further study the effects of keypoint reduction, we vary the threshold levels between the query and test images; thereby varying the quantity of indexed keypoints and query keypoints. This experiment allows us to determine whether using more keypoints for the query (on a smaller number of indexed keypoints) yields higher effectiveness. Table 2, shows the retrieval effectiveness of 200 queries on Collection  $C_1$  using three different variations for thresholding the query and test images, mainly 1000, 100, and 10; default denotes the original number of keypoints (1900 on average). Columns 1 and 2 indicate threshold value used for the test (indexed), and query images, respectively. The last column indicates the average candidate keypoints that are retrieved from the LSH index during query evaluation.

As shown in this table, using the original number of keypoints to query for images using threshold value of 100 yields a high average recall and precision, of 94.5% and 98.4%, respectively, with a little under 5 seconds; this is similarly observed for the use of threshold value of 1000 for the query image, taking on average close to 3 seconds. Also note that the number of candidate keypoints that are examined are reduced considerably with threshold value of 100; from a set of 62,549 to less than 10,000. This is an important finding, given that with this threshold value, the loss of average recall is minimal while the savings in terms of index size and query run-time is significant. Note that similar trends are observed for other threshold values.

Finally, to evaluate the effectiveness of our approach on large collections, we experiment with a threshold value of 100 on Collection  $C_2$  — containing 100,000 images. As shown in Table 3, the average recall and precision are similar to those of Collection  $C_1$ . However, the average query run-time has increased considerably with default and threshold value of 1000 in the query images, as a result of a larger candidate keypoint set. In contrast, the threshold value of 100 for the query images is relatively unaffected judging by the slight growth in candidate keypoints. Furthermore, this threshold parameter requires only 6.2 seconds, indicating that it scales well for large image collections. We do not experiment on the original approach on Collection  $C_2$  as the query run-time is impractical. Using this threshold value, we also observe an index size of 5.3 GB as compared to the estimated 76.5 GB using the original approach.

## 7 Conclusion

In this work, we have shown that the SIFT interest keypoints can be significantly pruned to reduce the amount of features that are indexed. Such pruning results in large reductions in both on-disk memory requirements and query evaluation time with only a slight loss in effectiveness.

To demonstrate the robustness of this approach,

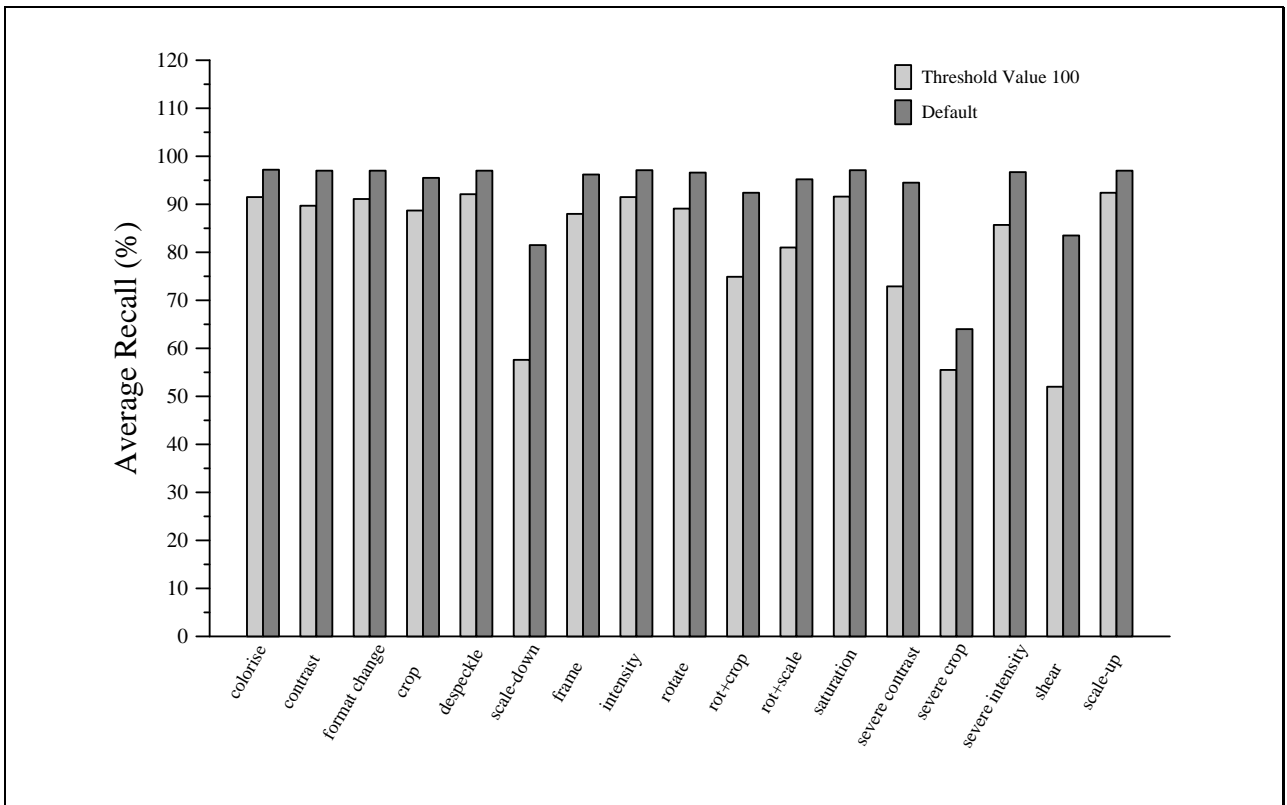


Figure 5: Average recall of each alteration over 20 queries — on Collection  $C_1$  — to retrieve other altered images using the original approach and threshold value of 100; each of the 17 alteration groups are described in Section 5, wherein there are 50 individual alterations in total.

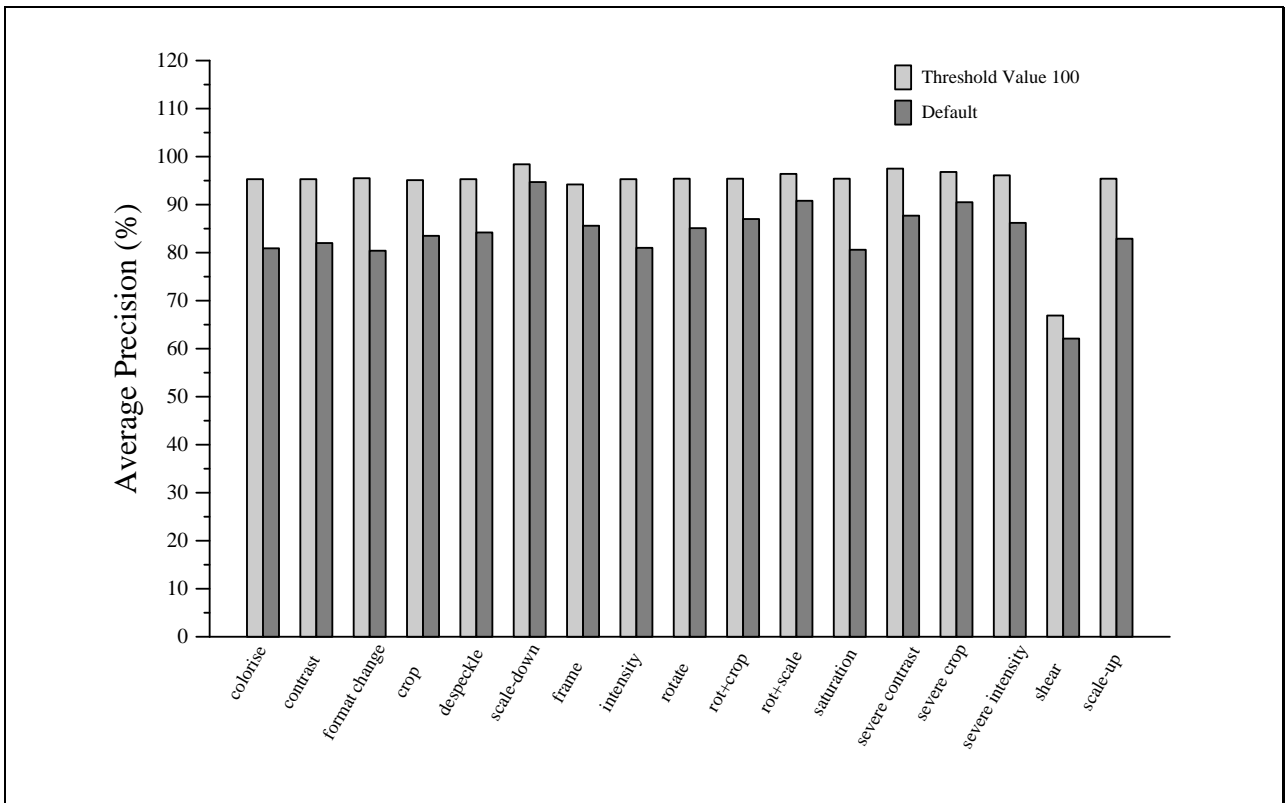


Figure 6: Average precision of each alteration over 20 queries — on Collection  $C_1$  — to retrieve other altered images using the original approach and threshold value of 100; each of the 17 alteration groups are described in Section 5, wherein there are 50 individual alterations in total.

Table 3: Average recall and precision (%) of 200 queries on Collection  $C_2$  using threshold value of 100. Column 2 indicates the threshold value used for the query images.

Index KP	Query KP	Avg. Recall	Avg. Prec.	Avg. run- time	Cand- idate Set
100	Default	94.5	98.2	40.8	19,215
	1000	94.5	98.4	30.7	10,728
	100	90.9	99.3	6.2	3,093
	10	56.9	100.0	0.5	367

we examine the effects of pruning on several kinds — including relatively severe ones — of image transformations. We show that our approach performs pleasingly well, with average recall close to the original approach; we also observe average precision to surpass the original approach.

Using parameters that show the best trade-off, our pruning method reduces the index size to approximately 1/10, and the query run-time to 1/50 of the original approach. Additionally, we demonstrate that, unlike the original approach, our method scales well for a large collection of 100,000 images.

## 8 Acknowledgment

This work was supported by the Australian Research Council.

## References

- Bawa, M., Condie, T. & Ganesan, P. 2005, LSH forest: Self-tuning indexes for similarity search., in A. Ellis & T. Hagino, eds, 'WWW', ACM, pp. 651–660.
- Chang, E., Wang, J. Z. & Wiederhold, G. 1998, RIME: A replicated image detector for the world-wide web, in 'Proc. SPIE Int. Conf. on Multimedia Storage and Archiving Systems III'.
- Datar, M., Immorlica, N., Indyk, P. & Mirrokni, V. S. 2004, Locality-sensitive hashing scheme based on p-stable distributions., in J. Snoeyink & J.-D. Boissonnat, eds, 'Symposium on Computational Geometry', ACM, pp. 253–262.
- Fischler, M. A. & Bolles, R. C. 1981, 'Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography.', *Commun. ACM* **24**(6), 381–395.
- Gionis, A., Indyk, P. & Motwani, R. 1999, Similarity search in high dimensions via hashing, in 'Proc. VLDB Int. Conf. on Very Large Data Bases', Morgan Kaufmann, Edinburgh, Scotland, UK, pp. 518–529.
- Grauman, K. & Darrell, T. 2005, Efficient image matching with distributions of local invariant features., in 'Proc. CVPR Int. Conf. on Computer Vision and Pattern Recognition', pp. 627–634.
- Hartung, F. & Kutter, M. 1999, 'Multimedia watermarking techniques', *Proceedings IEEE (USA)* **87**(7), 1079–1107.
- Indyk, P. & Motwani, R. 1998, Approximate nearest neighbors: Towards removing the curse of dimensionality., in 'Proc. STOC Int. Conf. on Theory of Computing', ACM Press, Dallas, Texas, USA, pp. 604–613.
- Jaimes, A., Chang, S.-F. & Loui, A. C. 2002, Duplicate detection in consumer photography and news video., in 'Proc. MM Int. Conf. on Multimedia', pp. 423–424.
- Johnson, N. F., Duric, Z. & Jajodia, S. 1999, On "Fingerprinting" images for recognition, in 'Proc. MIS Int. Workshop on Multimedia Information Systems', Indian Wells, California, pp. 4–11.
- Joho, H. & Sanderson, M. 2004, 'The spirit collection: an overview of a large web collection.', *SIGIR Forum* **38**(2), 57–61.
- Kang, X., Huang, J. & Shi, Y. Q. 2002, An image watermarking algorithm robust to geometric distortion., in 'Proc. IWDW Int. Workshop on Digital Watermarking', Springer, Seoul, Korea, pp. 212–223.
- Kang, X., Huang, J., Shi, Y. Q. & Lin, Y. 2003, 'A DWT-DFT composite watermarking scheme robust to both affine transform and JPEG compression.', *IEEE Trans. Circuits and Systems for Video Technology* **13**(8), 776–786.
- Ke, Y. & Sukthankar, R. 2004, PCA-sift: A more distinctive representation for local image descriptors., in 'Proc. CVPR Int. Conf. on Computer Vision and Pattern Recognition', IEEE Computer Society, Washington, DC, USA, pp. 506–513.
- Ke, Y., Sukthankar, R. & Huston, L. 2004, An efficient parts-based near-duplicate and sub-image retrieval system, in 'Proc. MM Int. Conf. on Multimedia', ACM Press, New York, NY, USA, pp. 869–876.
- Lowe, D. G. 2004, 'Distinctive image features from scale-invariant keypoints.', *Int. Journal of Computer Vision* **60**(2), 91–110.
- Lu, C.-S. & Hsu, C.-Y. 2005, 'Geometric distortion-resilient image hashing scheme and its applications on copy detection and authentication.', *Multimedia Systems* **11**(2), 159–173.
- Luo, J. & Nascimento, M. A. 2003, Content based sub-image retrieval via hierarchical tree matching., in 'Proc. MMDDB Int. Workshop on Multimedia Databases', pp. 63–69.
- Mikolajczyk, K. & Schmid, C. 2003, A performance evaluation of local descriptors., in 'Proc. CVPR Int. Conf. on Computer Vision and Pattern Recognition', pp. 257–263.
- Qamra, A., Meng, Y. & Chang, E. Y. 2005, 'Enhanced perceptual distance functions and indexing for image replica recognition.', *IEEE Trans. Pattern Analysis and Machine Intelligence* **27**(3), 379–391.
- Sebe, N., Lew, M. S. & Huijismans, D. P. 1999, Multi-scale sub-image search., in 'Proc. MM Int. Conf. on Multimedia', ACM Press, Orlando, FL, USA, pp. 79–82.
- Smeulders, A. W. M., Worring, M., Santini, S., Gupta, A. & Jain, R. 2000, 'Content-based image retrieval at the end of the early years', *IEEE Trans on Pattern Analysis and Machine Intelligence* **22**(12), 1349–1380.
- Zhang, D. & Chang, S.-F. 2004, Detecting image near-duplicate by stochastic attributed relational graph matching with learning., in 'Proc. MM Int. Conf. on Multimedia', pp. 877–884.