High-Confidence Near-Duplicate Image Detection

Wei Dong *
Independent Researcher
744 Eagle Ave, Ann Arbor MI 48103
wdong@wdong.org

Zhe Wang Moses Charikar Kai Li
Princeton University
35 Olden Street, Princeton NJ 08540
{zhewang, moses, li}@cs.princeton.edu

ABSTRACT

In this paper, we propose two techniques for near-duplicate image detection at high confidence and large scale. First, we show that entropy-based filtering eliminates ambiguous SIFT features that cause most of the false positives, and enables claiming near-duplicity with a single match of the retained high-quality features. Second, we show that graph cut can be used for query expansion with a duplicity graph computed offline to substantially improve search quality. Evaluation with web images show that when combined with sketch embedding [6], our methods achieve false positive rate orders of magnitude lower than the standard visual word approach. We demonstrate the proposed techniques with a large-scale image search engine which, using indexing data structure offline computed with a Hadoop cluster, is capable of serving more than 50 million web images with a single commodity server.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]

General Terms

Algorithms, Design

Keywords

near-duplicate, query expansion, entropy, graph cut

1. INTRODUCTION

Near-duplicate image detection is the task of finding different versions of the same image, i.e., images that are not exact duplicates in binary form, but can be visually identified as the same image having undergone various editing steps such as color mapping, scaling, format changing, etc. In particular, we are also interested in detecting partial-duplicates, i.e., images with padding and cropping, but still containing sub-images that are near-duplicates.

*This work was done while the author was a graduate student in Princeton University.

ICMR '12, June 5-8, Hong Kong, China

Near-duplicates are common in web images. Being able to efficiently detect them is important to applications such as copyright violation detection and finding alternate versions of existing images. Eliminating near-duplicates is also an important preprocessing step for many applications.

Previous researches have achieved high quality search results in small to medium scale (one million images or less) experiments [10, 14, 3, 21], but the challenge remains in pushing the retrieval scale to the World Wide Web level. Today's web contains hundreds of billions of images and continues to grow rapidly. Both commercial search engines and academic experiments run at a much smaller scale. For example, Tineye (`http://www.tineye.com/`), the first commercial near-duplicate image search engine available to the public, indexes 1.8 billion images as of 2010. In the literature, a recent system [21] achieved 1.9 second search time on one million images with a single machine.

The goal of this paper is to design a building block for a large-scale near-duplicate image search engine — a single node system that substantially improves the search capacity over existing approaches without increasing search time, and most importantly, conducts search with high confidence, i.e., low false positive rate. The false positive problem is not prominent in traditional content-based image search due to the lack of an objective and unambiguous definition of visual similarity. In our case, there is a clear yes-or-no answer to whether two images are near-duplicates, and one can easily spot irrelevant images in the search result, so it is important for the search result not to be dominated by false positives. The challenge is that there is usually only a few (a few to thousands) true positives, and when multiplied by a virtually unlimited number of background images (e.g., 1 billion), even a very small false positive rate (e.g., 0.1%) could potentially generate an overwhelming number of false positives.

Even though various approaches have been published, they all fail to meet our efficiency and/or quality requirements. Global features like GIST [7], including pooled local features [5, 9], are not intended for detecting partial-duplicates, and it is essential to index local features. SIFT [11] has been firmly established in the literature as the local feature of choice for a variety of image retrieval tasks, including near-duplicate image detection, we focus on system designs using SIFT features.

The state-of-the-art approach to near-duplicate image detection is the Bag-of-Word (BoW) model, i.e., to convert SIFT features into visual words via vector quantization, and then apply well-developed text retrieval techniques like inverted index and TF-IDF ranking [14, 15, 21]. This approach, however, is not sufficient to lower false positive rate to our desired level because: first, as we will show later, a small fraction of SIFT features are intrinsically ambiguous and can cause a large amount of false positives; second,

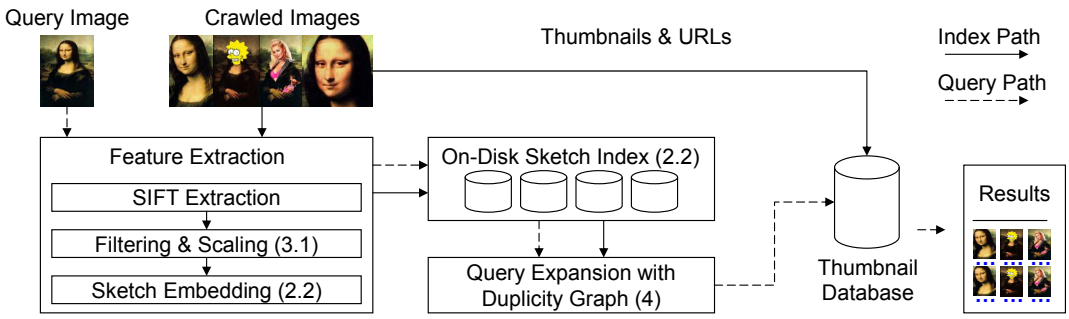

Figure 1: System architecture. Section numbers are indicated for the key components.

vector quantization loses the discriminating power of raw SIFT features [2] and further increases false positive rate. We will show by experiments that TF-IDF ranking and geometric verification have their limits on compensating for the information lost during vector quantization.

Another approach is to conduct retrieval directly on raw SIFT features with Locality Sensitive Hashing [8, 10, 3]. The problem is that the large raw feature size (typically a few hundreds of 128-dimension feature vectors per image) imposes a high storage and retrieval cost and could become a capacity bottleneck. It is known that plain LSH takes too many hash tables [12], and the recently developed Multi-Probe LSH [12], although reduces storage cost, requires too many disk seeks when implemented out of core. Expensive geometric verification is also needed for this approach to ensure low false positive rate.

In this paper, we achieve both efficiency and quality by making the following three contributions.

First, we eliminate the previously indispensable stage of geometric verification by making the following important observation: false positives are mainly caused by a small fraction of SIFT features that are intrinsically ambiguous due to the lack of internal structure within the SIFT region. We propose an entropy-based filtering method that can effectively filter out these non-discriminative features, and show that with the retained high quality features, *a single match is sufficient for claiming a near-duplicate relationship between images with high confidence*. This finding dramatically simplifies our retrieval model and system design and also reduces the number of SIFT features to be indexed.

Second, we propose a query expansion method based on graph cut. We construct offline a duplicity graph by connecting all pairs of images predicted to be near-duplicates. At query time, all vertices reachable from the initial search results are potentially positive results. We use a graph cut method to exclude pathetic cases when large clusters of negative images are connected to the initial results. Our query expansion method substantially improves recall, or equivalently, reduces false positive rate by orders of magnitude at the same recall level.

Third, we implemented a search system based on the proposed techniques. For space and time efficiency, we represent SIFT features with sketches [6], which is a bit-vector representation more compact than raw feature vectors and more accurate than visual words, and index the sketches with an efficient out-of-core data structure [13]. Our system is capable of indexing more than 50 million web images on a commodity server and returning search results in less than two seconds.

We evaluated our system as well as some representative existing techniques with a benchmark (will be made public upon the acceptance of this paper) consists of ten thousand manually verified near-duplicate web images and one million random background images. Our system achieves a false positive rate of 2.4×10^{-6} (at recall 0.8), three orders of magnitude smaller than the state of the art [21].

2. SYSTEM DESIGN

2.1 Overview

Our system is based on a very simple retrieval model. Each image is represented as an ID with an attached set of features: $I_i = \langle i, \{f_{ij}\}\rangle$. A flat database DB $= \{\langle f_{ij}, i\rangle\}$ of ID-feature pairs is maintained for the indexed image dataset. For each query image $Q = \langle \cdot, \{f_k\}\rangle$, a query is issued for each feature f_k to retrieve database records whose features are within a fixed distance threshold d:

$$N_d(f_k) = \{\langle f_{ij}, i\rangle \in \text{DB} | \, \|f_k - f_{ij}\|_2 \leq d\}.$$

The retrieved lists are merged: $N_d(Q) = \cup_k N_d(f_k)$, and all IDs in $N_d(Q)$ are returned as near-duplicates. An optional query expansion stage (Section 3.1) can be applied to improve recall.

Local features can get matched to different degrees and there can be multiple matched local features, with consistent or inconsistent spatial layout. It is non-trivial to design a principle measure to cover all these aspects. We circumvent the complexity by assuming that we can produce local features of sufficiently high quality, so that a single match of local feature provides a strong indication of image duplicity.

An extra ranking stage can be potentially carried out to re-arrange the search results with voting or geometric verification — if geometric information are retained with SIFT features. However, this is not critical for our system because of our extremely low false positive rate, and is not the interest of this paper.

The system architecture is shown in Figure 1, with Section numbers indicated for the major components. The sketch embedding and indexing methods are based on existing techniques and are briefly described in the remainder of this section. The novel feature filtering and query expansion techniques are detailed in the following two sections.

2.2 Sketch Construction and Indexing

We use sketch embedding, a compact bit-vector representation, of raw SIFT features to achieve space and time efficiency, without losing as much as accuracy as visual word representation. Specifically, we use the method proposed by Dong et al. [6] and implemented in the open-source LSHKIT [1]. It does not require offline

[1] `http://lshkit.sourceforge.net/`

Figure 2: Illustration of SIFT features. A random sample of SIFT features extracted from the Lena image is plotted. Each SIFT feature consists of 4×4 histograms of edge orientation. In general, a matching of SIFT features means that the spatial configuration of 16 small regions is consistent between two images. It is a strong signal of near-duplicity, except for features like A and D, which contain little internal structure. We use entropy-based filtering to remove such bad features.

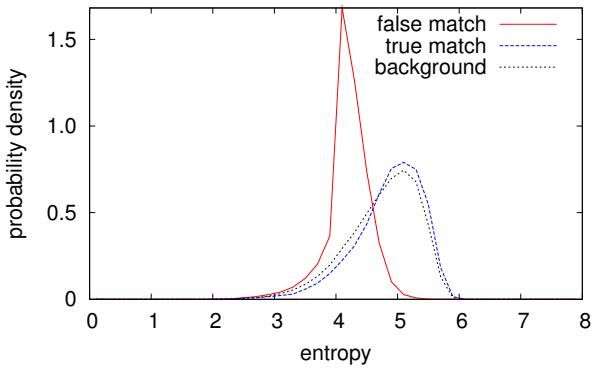

Figure 3: Entropy distributions of SIFT features that are sample from (1) false matches, (2) true matches, (3) random background. Compared to the others, the entropy distribution of wrongly matched SIFT feature is narrower, and the peak is at a smaller value.

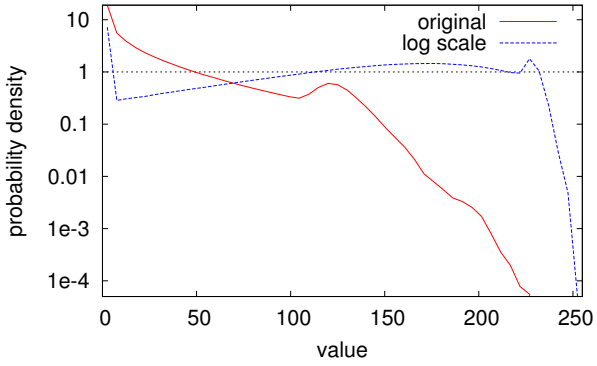

Figure 4: Distribution of SIFT feature values before and after log scaling. Log scaling makes the curve much more flat.

training and can be tuned to be sensitive to a particular distance range, which fits our distance thresholding retrieval model.

For a point $p \in \mathbb{R}^d$, we define the following function to map it to a single bit:

$$f(p) = \lfloor \frac{A \cdot p + b}{W} \rfloor \mod 2$$

where $A \in \mathbb{R}^d$ is a random vector with each dimension sampled independently from the standard Gaussian distribution, and $b \in \mathbb{R}$ is a random variable sampled from the uniform distribution $U[0, W)$. W is called the window size, which controls the distance range that the sketch is sensitive to. We use $W = 8.0$ in our system. To obtain a B-bit sketch, we independently sample B such random mappings with equal W, and concatenate the produced bits. We find that for our system, $B = 128$ is an appropriate size which both retains good accuracy and provides an $8\times$ compression over raw SIFT features. We claim a match if the hamming distance is ≤ 3 for online search, or ≤ 2 for offline duplicity graph construction.

Our indexing data structure for online search is a simple special case of the technique proposed by Manku *et al* [13]. We divide the 128-bit into 4 blocks, and build a hash table with each block as a key. For a pair of matching sketches, there could be at most 3 different bits, so at least one hash key is identical. By searching all the 4 hash tables, we are guaranteed not to miss any matching sketches. Offline duplicity graph construction is carried out with the same technique. The sketch index and the duplicity graph are constructed with Hadoop on a cluster, but after being created, the data structures are compact enough to be served with a single server.

SIFT feature indexing is orthogonal to the main techniques proposed in this paper, and can be easily replaced with other methods. For example, Weiss et al. [20] proposed a sketch construction method based on offline learning.

3. FEATURE PROCESSING

The wide adoption of the visual word approach creates the impression that SIFT is a point feature (hence the phrase key point), and that a strong match between two images probably needs the agreement of multiple matches of key points. But in fact, as we can see from Figure 2, a SIFT feature describes a region that usually occupies a significant area in the image and has rich internal content. We believe that in most cases, one matched SIFT region, like B and C in Figure 2, is already a strong indication of near-duplicity. Our retrieval model is based on a simple thesis: SIFT feature match $\Rightarrow$ image match. We will show with experiments that this seemingly loose assumption works sufficiently well with high quality SIFT features produced with the two techniques described in this section.

We use VLFeat[2] [19] for SIFT feature extraction. To limit the number of features generated from each image, we resize large images to 300×300. After applying entropy-based filtering described below, roughly 80 features are extracted from each image.

Entropy-Based Filtering

The SIFT features best suitable for near-duplicate detection are those with rich internal structure. SIFT features associated with near-empty regions are the main source of false positives: they tend to occur frequently and get easily matched against one another. We use entropy to measure the richness of internal structure of a SIFT region. Specifically, we treat the SIFT feature $F =$

[2] `http://www.vlfeat.org/`

$[f_1, f_2, \ldots, f_{128}]$ as 128 samples of a discrete random variable in $\{0, 1, \ldots, 255\}$:

$$H(F) = -\sum_{i=0}^{255} p_i(F) \log_2 p_i(F), \qquad p_i(F) = \frac{|\{k \mid f_k = i\}|}{128}.$$

Note that each SIFT dimension is a 8-bit integer, so the entropy has a range of $[0, 8]$.

We measure the entropy distributions of (1) SIFT features generating false matches, (2) SIFT features generating true matches and (3) randomly sampled SIFT features as background. The results are plotted in Figure 3. We see that the features causing false positives have a more concentrated distribution, with the peak at a relatively low entropy value of 4.1, while the good and background SIFT features have very similar distributions, both have a peak at 5.1. In our system, we discard all SIFT features with an entropy smaller than 4.4.

Log Scaling

The values for individual dimensions of SIFT feature follow a near-exponential distribution, with small values dominating the whole distribution. A SIFT value has a range of $[0, 255]$, but find more than 97% of the values are smaller than 128 and 67% of the values are smaller than 25. This means that the range of the value is not efficiently used. We propose to scale the dimensions of SIFT features logarithmically. Figure 4 shows the SIFT value distribution before and after log scaling. Note that the distribution is more uniform after log scaling.

3.1 Query Expansion with Graph Cut

We set a tight threshold on similarity measure to ensure a low false positive rate, but that also cause us to fail to detect a significant portion of true positives. Query expansion can be used to recover some of the missed positive images. In this section, we propose an effective method of query expansion that particularly fits our retrieval model.

The high level idea of query expansion is to use the search results to query the database again. The procedure can be recursively applied until no new results are found (transitive closure expansion). As the queries used in expansion are all images known to the system, we can accelerate the process by computing the near-duplicates of each indexed image offline.

To ease further discussion, we restate the above in the language of graph theory. We use a *duplicity graph* G to represent the offline computed duplicity relationship. Each indexed image is represented as a vertex in G, and an edge is established between each pair of images with at least one matched local feature. We use $\overline{G}$ to represent G augmented with the query image q and edges between the query and the initial search results. We use $B_k(q)$ to denote the set of vertices that can be reached from q within k steps, so $B_1(q)$ is exactly the initial search results without query expansion. We use $B_\infty(q)$ to denote the connected component of $\overline{G}$ including q, or equally, the results one will get with transitive closure expansion. $B_k(q)$ and $B_\infty(q)$ consists of our baseline query expansion strategies.

To inspire our improved strategy, we categorize potential sources of false positives involved in query expansion into the following three types:

Sporadic False Positives: This is also how false positives get into the initial search results, and is governed by the false positive rate.

Clusters: Dense clusters exist in G. They could either be meaningful clusters of popular images like *Mona Lisa*, or clusters of unrelated noisy images. One common cause of the latter is that images with dense patterns and rich in local feature, e.g., windows of a skyscraper, tend to get matched to a lot of false positives. Such "hub" images can even interconnect to create huge noise clusters. No matter how a cluster is generated, it could dramatically increase false positive rate when hit via a sporadic false positives.

Bridging Images: Even if we achieve 0 false positive rate in building G, false positives could still occur in query expansion via bridging images. For example, an image composed of both *Mona Lisa* and *The Last Supper* should be connected to both clusters by definition. When the query contains only *Mona Lisa*, transitive closure expansion will reach the cluster of *The Last Supper* via the bridging image.

Because our graph has extremely low false positive rate, we are relatively confident when query expansion leads to individual images only. What we want to avoid is hitting an irrelevant cluster through either sporadic false positives or bridging images. On the other hand, if a cluster has many different connections to the query, then we have reasons to believe that they are actually positive results. In all, we want the query expansion result $B(q)$ to be connected, have many internal connections but few connections to the rest of $\overline{G}$, and be large. We find that these requirements can be jointly captured by the *conductance* of a subgraph, defined as

$$\Phi(S) = \frac{|E[S, \overline{G} \setminus S]| \quad \text{//connections to the rest of graph}}{\mu(S) \quad \text{//size and intra-connections}},$$

where S is the subgraph of interest, $E[S, \overline{G} \setminus S]$ is the set of edges between S and $\overline{G} \setminus S$, and $\mu[S]$, called the volume of S, is the sum of degrees of vertices in S[3]. The sparse cut problem is to optimize for minimal conductance:

$$B(q) = \arg \min_{\substack{S \subset \overline{G}, q \in S \\ S \text{ connected}}} \Phi(S).$$

The problem is hard, but efficient approximate solutions exist [17, 1]. In our system, we use an adapted version of PageRank-Nibble [1], which is listed in Algorithm 1. It simplifies the original algorithm by (1) removing many parameters that are necessary when the algorithm is used as a subroutine of a provable graph partitioning algorithm, and (2) choosing the sweep set with the smallest conductance rather than returning one only when it satisfies a number of requirements. The subroutine to compute approximate PageRank is exactly the same as in the original paper. The algorithm requires two parameters $\alpha, \epsilon \in (0, 1)$. We find the final results relatively independent on those parameters, and use $\alpha = 0.5$ and $\epsilon = 0.00001$ in our experiment. The computational cost of the method, assuming G fits in main memory, is negligible when compared to sketch index lookup, and the improvement in recall is huge: from 45% to nearly 80%.

The size of G depends on both the size of the dataset and how often duplicate images appear in the dataset. When we indexed a dataset including 30 million Flickr images and 30 million product images crawled from `taobao.com`, the size of G is 7.8GB and fits in the main memory of our server.

4. EXPERIMENTAL SETUP

We are interested in evaluating how the proposed techniques improve search quality. This section describes the configuration of our experiments, including the dataset, performance measures and

[3]This definition is equivalent to the standard definition [17] under the assumption that S is a always smaller than $\overline{G} \setminus S$. This assumption is reasonable in our setting as we are never to return more than half of the indexed images.

Algorithm 1: PageRank-Nibble

Data: graph $\overline{G}$ (adjacent list), query q, α, ϵ
begin
  $P \longleftarrow$ ApproximatePageRank(q, α, ϵ)
  // P consists of vertices with non-zero PageRank,
  // sorted in descending order of PageRank
  $e \longleftarrow 0$ // Count $|E[S, \overline{G} \setminus S]|$
  $v \longleftarrow 0$ // Count volume of S
  $\min \leftarrow +\infty$, $B \leftarrow \emptyset$ // Lowest conductance
  for $i \longleftarrow 1$ **to** $|P|$ **do**
    $v \longleftarrow v + |\overline{G}[P_i]|$
    $e \longleftarrow e + |\overline{G}[P_i] \setminus P_{1..i-1}| - |\overline{G}[P_i] \cap P_{1..i-1}|$
    $c \longleftarrow e/v$
    if $c < \min$ **then**
      $\min \longleftarrow c$
      $B \longleftarrow P_{1..i}$
  return B

evaluated methods. The results are to be reported in the next section.

4.1 Dataset

We gathered ground truth images by issuing text queries to the popular image search engines and then manually identifying near-duplicate images from the search results:

1. 81 titles of famous paintings, movies and CDs were picked;

2. Four popular image search engines, i.e., Google, Bing, Yahoo and Flickr, were queried with these titles, and original images were downloaded using the URLs returned by the image search engines;

3. Exact duplicates were removed from the crawled images using MD5 checksums. The remaining images were manually inspected, and between 40 and 120 near-duplicate images were identified from each group. The total dataset consists of 10839 images.

We did not use text information in our system so images gathered with text queries should allow us to obtain a good estimation of performance on real web images.

We use one million random Flickr images as background. Even though we have collected more web images, some methods we compare against (e.g. linear scan of raw features) are very expensive and do not scale up.

4.2 System Environment

We used commodity servers of the following configuration for experiments: dual quad core Intel E5430 2.66GHz CPU; 16GB main memory; 4 Seagate Barracuda 7200.11 1.5TB disks (for online serving, less than 30% of both main memory and disk storage are used). All machines ran CentOS 5.3 with Linux kernel 2.6.18.

4.3 Performance Measures

We are mainly interested in evaluating the receiver operating characteristic (ROC), which includes the following two measures.

True Positive Rate (Recall): number of retrieved near-duplicates divided by the total number of near-duplicates in the ground truth dataset.

False Positive Rate: number of falsely retrieved background images divided by the total number of background images.

When evaluating a retrieval system, a common practice is to use a single measure of quality, e.g., precision@k, or average precision. Such measures are meaningful only when the ratio between positive and negative is fixed, and are not appropriate in our setting because we gather positive and negative examples separately, and the combination does not necessarily approximate the real-life ratio, which is subject to change depending on the application.

When evaluating false positive rate, we assumed that there are no near-duplicates of the ground truth images in the one million background images. This might not be true as we did not label the background data, which would require a huge amount of human labor. However, we believe that this approximation is not a problem for our particular evaluation setup. First, the expected number of true positives in the background set is extremely small according to our experiments. We achieve a false positive rate of 4.9×10^{-7} at recall 0.43 (without query expansion). That means that the expected number of near-duplicates mixed in the one million background dataset is $4.9 \times 10^{-7}/0.43 \times 10^{6} = 0.1$, less than one. Second, even if there are near-duplicates in the background images, that only means that the real false positive rate of our method is lower (better) than what we report.

4.4 Simulation

We built a simulator to evaluate various retrieval methods. Our simulator is different from a full search engine in that it does not build an out-of-core index data structures but rather sequentially scans all the data images. All performance numbers reported are exactly the same as if full search engines were built, except that we cannot use the simulator to evaluate search speed.

The following retrieval methods are implemented in the simulator.

Raw The raw SIFT feature retrieval method as described in Section 2.1. This method provides the best search quality, but is too time-consuming for online search. It represents the highest search quality achievable with our retrieval model (without query expansion) when an ideal feature index is used.

TF-IDF The bag-of-word method with TF-IDF ranking. We use a dictionary of one million visual words and soft assign each feature to four visual words.

Bundle The TF-IDF method augmented with bundle information [21].

Sketch The proposed sketch based retrieval method.

Sketch+Exp The sketch based retrieval method with result set expansion.

5. EXPERIMENTAL RESULTS

We first evaluate the individual techniques proposed, then compare our full system to existing retrieval methods. Finally, we provide some space/time performance numbers of our system.

5.1 Entropy-Based Filtering and Log Scaling

Because entropy-based filtering and log scaling are independent from the retrieval models, we first show their improvement by applying them on the raw SIFT feature retrieval method. We then apply them to all the retrieval methods for fair comparison.

Figure 5 shows the huge impact these two feature processing steps can make. Entropy-based filtering helps to reduce false positive rate by orders of magnitude, which is especially prominent when true positive rate is low. Without entropy-based filtering, there are always 354 background hits no matter how we tighten the retrieval threshold. These potential false positives are caused by SIFT features that are intrinsically ambiguous, and would otherwise require geometric verification to get eliminated. With entropy-based filtering, false positive rate can be tuned very close to zero, which is the normal behavior desired by a retrieval method.

The effect of log scaling is not as gigantic. Unlike entropy-based filtering, it mainly works at the range of high true positive rate.

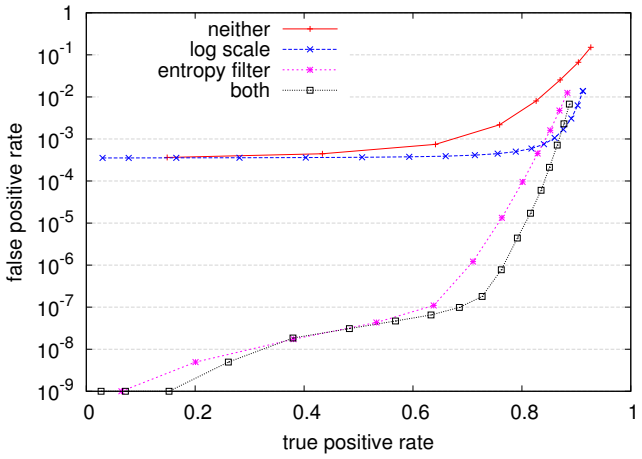

Figure 5: The effects of entropy-based filtering and log scaling. Entropy-based filtering eliminates most of the false positives caused by ambiguous SIFT features. Log scaling helps further reducing false positives in most cases.

Even when stacked with entropy-based filtering, it still helps to further lower the false positive rate.

5.2 Query Expansion

Figure 6 compares various strategies of query expansion. We observe that for the $B_k(q), k \geq 2$ family of strategies to achieve the same recall, false positive rate increases as k becomes larger, with $B_2(q)$ being the most practical strategy. This is because larger k means more aggressive query expansion, and is more susceptible to false positives. Our graph-cut-based expansion strategy is way better in avoiding false positives, achieving false positive rate two orders of magnitude lower than $B_2(q)$ at recall around 0.8.

5.3 Comparison of Retrieval Methods

Figure 7 shows the performance or various retrieval methods. We see that the raw SIFT feature retrieval method has the best performance, except that it is too time-consuming to be practically applicable. Plain TF-IDF ranking with visual words has the worst performance. Furthermore, the items of around 20,000 background images (indicated by the rightmost point on the curve) have to be first retrieved from disk before ranking can be carried out, i.e., the retrieval cost is the same for all points on the curve. By attaching geometric information on the visual words, the bundling method helps to reduce background hit rate by a factor of 3 for most points on the curve. The curve of our sketch method lies in the middle between the BoW methods and the raw SIFT method, and by applying query expansion, true positive rate can be significantly improved to make the performance very close to the raw SIFT curve.

In the real system, we use a hamming distance threshold of 3, which corresponds to true positive rate = 0.43 and false positive rate = 4.9×10^{-7} without query expansion, and true positive rate = 0.79 and false positive rate = 2.5×10^{-6} with query expansion. For the sketch curve, false positive rate increases exponentially with recall, and the point we choose represents a good trade-off between precision and recall.

5.4 Whole System Performance Numbers

We use our search engine to index 50 million web images. Table 5.4 summarizes the whole system performance numbers to-

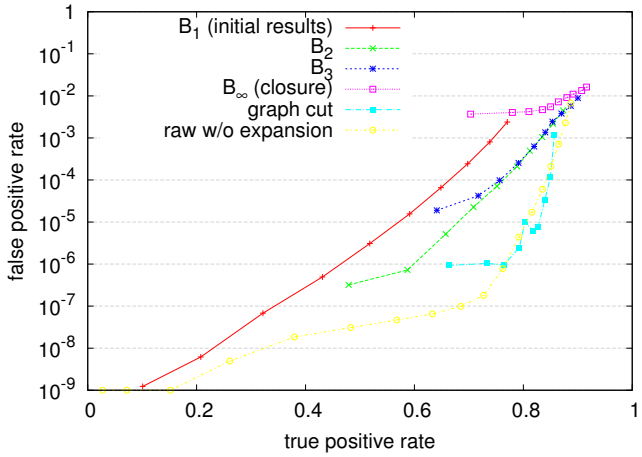

Figure 6: Comparison of various query expansion methods with sketch-based retrieval. Raw SIFT retrieval (impractical for online search due to high costs in time and space) performance is included for reference. Graph cut achieves the best accuracy among the sketch-based retrieval methods.

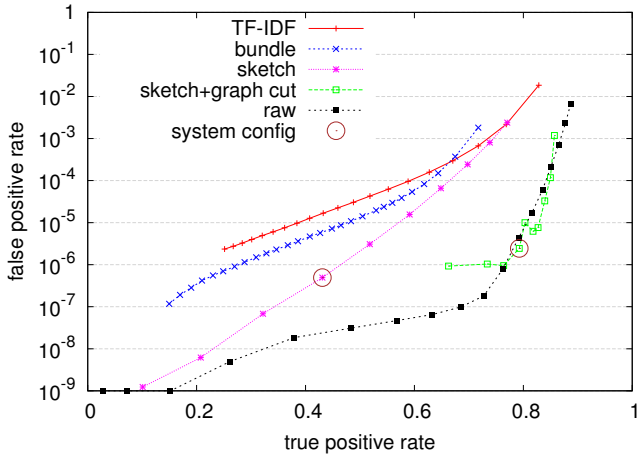

Figure 7: Comparison of various retrieval methods. At the recall level of our system configuration, the sketch-based method lowers the false positive rate of the bundling method by an order of magnitude. Recall can be improved by result set expansion.

Image Indexed	50 million
Offline Processing	$\sim$ 3 days $\times$ 66 nodes
Index memory usage	$16 \times 177MB = 2.8GB$
Index disk usage	$16 \times 81GB = 1.3TB$
Thumbnails (128×128) & URLs	$374GB$
Overall memory usage	$3.2GB$
Overall disk usage	$1.7TB$
Average CPU usage§	1.8%
Peak CPU usage§	3.8%
Response time †	1.29 ± 1.02 seconds
False positive rate‡	4.9×10^{-7}
Initial true positive rate‡	0.45
True positive rate w/ Expansion‡	0.70

Table 1: System performance numbers. §Running one query at a time. †Feature extraction and index search, without retrieving thumbnails. ‡Estimated with one million samples.

gether with the true/false positive rates estimated with small scaled experiments to be reported later. Our sketch index runs one sketch query for 0.069 second sequentially, and for our benchmark, each image requires 77.7 sketch queries on average. To reduce the search time, we scatter the index hash tables to four disks for parallel reading, and batch process the sketch queries so disk accesses can be properly scheduled [10]. The result is that feature extraction and sketch searching can be finished in 1.29 seconds on average. That is faster than the 1.9 second reported by the bundled feature paper [21] to search one million images.

Our sketch index takes 1.3TB disk storage and 2.8GB main memory. The duplicity graph for query expansion takes about 5GB main memory. The thumbnails ($128x128$), densely stored in a single file, takes about 400GB disk space. Overall, less then 30% of the server's memory and disk storage is used. We use a cluster of 66 nodes for offline indexing, and processing 50 million images takes about one day.

6. RELATED WORKS

Near-duplicate image detection, especially detecting duplicates subject to cropping and padding, has only been practically viable after the invention of the groundbreaking SIFT local feature [11]. Ke et al. [10] were the first to study a SIFT based method for near-duplicate image detection. They used LSH to index PCA-SIFT (a variant of SIFT) and RANSAC for verification, and demonstrated superior retrieving performance at the scale of 20K images. They showed that running queries in a batch and linearizing disk accesses can bring a $20\times$ speed-up. We also implemented similar disk scheduling algorithm in our system.

Most recent methods are based on the Bag-of-Word model [14, 15, 21]. Although the BoW approach has been successful at million image level, it has intrinsic limitations preventing it from scaling up to the level of a billion images. On one hand, a large visual dictionary is needed for better discriminating power and to limit the amount of data needed to be read from disk. On the other hand, the larger dictionary usually means lower recall when visually identical local features are over-quantized into different visual words. Soft assignment [16] can be used to recover some of the lost recall due to over-quantization, but it also enlarges the retrieval cost by several times. One million is a widely accepted size for a visual dictionary [15, 21]. Such a large dictionary can be efficiently created by hierarchical clustering [14]. Assume we extract 100 features from each image on average, and these features are uniformly distributed across all the visual words. The chance for any two images to share at least one visual word is about 0.01. That means nearly 1% of all indexed images will be ranked for each query. When we need to index tens of millions of images on a single server, the retrieval and ranking cost would be too high to make a online search system possible.

Several approaches have been explored to improve the discriminating power of visual words. The visual phrase [23] method links near-by visual words to form phrases and conduct retrieval at phrase level. Visual phrases do have much higher discriminating power than visual words. However the number of visual phrases extracted from each image can easily grow out of control. Bundled Feature [21] is another method to improve the discriminating power of visual words. It attaches information on its relative position within its surrounding MSER region to each visual word. Thus explicit geometric verification can be avoided. However, the disk I/O problem is not solved because retrieval is still carried out at individual visual word level.

Turcot and Lowe [18] proposed a SIFT feature pruning method based on the observation that a SIFT feature is only useful when it contributes to a matching within the database, assuming that each image in the database has at least one near-duplicate. This strategy is effective in reducing the number of features by more than an order of magnitude without compromising search quality, and its application is not limited to near-duplicate detection. The drawback is that, effectively only images with duplicates in the database are indexed, which is not usually tolerable. Also, this method aims at reducing the space overhead, rather than filtering out low-quality features. Our entropy-based filtering method is orthogonal to this methods and can be used as a complement.

Chum et al. [4] studied various query expansion methods, including transitive closure expansion, for local-feature based object retrieval with vector quantization and geometric verification. The other methods proposed, like average query expansion and multiple image resolution expansion are potentially applicable to near-duplicate image detection, but involves reconstructing queries unknown to the system, and running those new queries could substantially increase search time.

Xu et al. [22] proposed to divide images into overlapping and non-overlapping blocks over multiple levels, and to compare and align the blocks with SIFT features and Earth Mover's Distance(EMD). Even though the method is reported to achieve high detection quality, it is only applicable to small datasets as it involves a very expensive similarity measure and requires scanning the full dataset in brute force.

7. CONCLUSION

We proposed two techniques for large-scale near-duplicate image detection:

- We found that entropy-based filtering can eliminate $> 99\%$ of false positives, and allows one to claim near-duplication relationship with a single match of the retained high quality features.

- We developed a query expansion method with graph cut to substantially improve search quality.

With the proposed techniques, we built a search engine that is capable of online serving more than 50 million web images with a single commodity server and achieving a false positive rate three orders of magnitude smaller than the standard visual word approach.

8. REFERENCES

[1] R. Andersen, F. Chung, and K. Lang. Local graph partitioning using pagerank vectors. *FOCS*, 2006.

[2] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *CVPR*, 2008.

[3] O. Chum, J. Philbin, M. Isard, and A. Zisserman. Scalable near identical image and shot detection. In *CIVR*, 2007.

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

[5] W. Dong, M. Charikar, and K. Li. Efficiently matching sets of features with random histograms. In *MM'08: Proceedings of the 16th ACM International Conference on Multimedia*, Vancouver, Canada, 2008.

[6] W. Dong, M. Charikar, and K. Li. High dimensional similarity search with sketches. In *SIGIR*, 2008.

[7] M. Douze, H. Jégou, H. Sandhawalia, L. Amsaleg, and C. Schmid. Evaluation of GIST descriptors for web-scale image search. In *Proceeding of the ACM International Conference on Image and Video Retrieval*, CIVR '09, pages 19:1–19:8. ACM, 2009.

[8] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC*, 1998.

[9] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *IEEE Conference on Computer Vision & Pattern Recognition*, pages 3304–3311, jun 2010.

[10] Y. Ke, R. Sukthankar, and L. Huston. An efficient parts-based near-duplicate and sub-image retrieval system. In *ACM MM*, 2004.

[11] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[12] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe lsh: efficient indexing for high-dimensional similarity search. In *VLDB*, 2007.

[13] G. S. Manku, A. Jain, and A. D. Sarma. Detecting near-duplicates for web crawling. In *WWW*, 2007.

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

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

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

[17] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC*, 2004.

[18] P. Turcot and D. Lowe. Better matching with fewer features: The selection of useful features in large database recognition problems. In *ICCV Workshop on Emergent Issues in Large Amounts of Visual Data*, 2009.

[19] A. Vedaldi and B. Fulkerson. Vlfeat – an open and portable library of computer vision algorithms. In *Proceedings of the 18th annual ACM international conference on Multimedia*, 2010.

[20] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*. 2009.

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

[22] D. Xu, T.-J. Cham, S. Yan, and S.-F. Chang. Near duplicate image identification with patially aligned pyramid matching. In *CVPR*, 2008.

[23] S. Zhang, Q. Tian, G. Hua, Q. Huang, and S. Li. Descriptive visual words and visual phrases for image applications. In *ACM MM*, 2009.