Computer Vision and Image Understanding 124 (2014) 31–41

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Fast and accurate near-duplicate image search with affinity propagation on the ImageWeb

Lingxi Xie[a,b,c,*], Qi Tian[d], Wengang Zhou[e], Bo Zhang[a,b,c]

[a] State Key Laboratory of Intelligent Technology and Systems (LITS), Tsinghua University, Beijing 100084, China
[b] Tsinghua National Laboratory for Information Science and Technology (TNList), Tsinghua University, Beijing 100084, China
[c] Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
[d] Department of Computer Science, University of Texas at San Antonio, TX 78249, USA
[e] Electronic Engineering and Information Science Department, University of Science and Technology of China, Hefei 230027, China

ARTICLE INFO

Article history:
Received 18 August 2013
Revised 29 November 2013
Accepted 24 December 2013

Keywords:
Large-scale image search
ImageWeb
Graph propagation
Search process tradeoff

ABSTRACT

Near-duplicate image search in very large Web databases has been a hot topic in recent years. In the traditional methods, the Bag-of-Visual-Words (BoVW) model and the inverted index structure are very widely adopted. Despite the simplicity, efficiency and scalability, these algorithms highly depends on the accurate matching of local features. However, there are many reasons in real applications that limit the descriptive power of low-level features, and therefore cause the search results suffer from unsatisfied precision and recall. To overcome these shortcomings, it is reasonable to re-rank the initial search results using some post-processing approaches, such as spatial verification, query expansion and diffusion-based algorithms.

In this paper, we investigate the re-ranking problem from a graph-based perspective. We construct ImageWeb, a sparse graph consisting of all the images in the database, in which two images are connected if and only if one is ranked among the top of another's initial search result. Based on the ImageWeb, we use HITS, a query-dependent algorithm to re-rank the images according to the affinity values. We verify that it is possible to discover the nature of image relationships for search result refinement without using any handcrafted methods such as spatial verification. We also consider some tradeoff strategies to intuitively guide the selection of searching parameters. Experiments are conducted on the large-scale image datasets with more than one million images. Our algorithm achieves the state-of-the-art search performance with very fast speed at the online stages.

1. Introduction

With more than twenty years' efforts, content-based image retrieval (CBIR) has become a successful application in computer vision. It provides an effective way of bridging the intent gap by analyzing the actual contents of the query image, rather than the metadata such as keywords, tags, and/or descriptions associated with the image. With compact image representation, it is possible for the state-of-the-art Web image search engines such as Google and Bing to handle billions of images and process each query with real-time response.

To search among a large corpus of images, the Bag-of-Visual-Words (BoVW) model [1] is widely adopted. The BoVW-based image search framework contains two major stages, *i.e.*, offline indexing and online searching. At the offline stage, local descriptors [2] are extracted on the crawled images, quantized onto a large visual vocabulary [3], and indexed into the inverted structure [1]. At the online stage, local descriptors are also extracted on the query image, and quantized into visual words to access the corresponding entries in the inverted index. Finally, all the retrieved inverted lists are aggregated as ranked search result.

Despite the simplicity, efficiency and scalability of the BoVW-based image search framework, the search results often suffer from the unsatisfied precision and recall. The main reasons arise from the limited descriptive power of low-level descriptors and the considerable information loss in the quantization step. In fact, the accurate matching between local features could be highly unstable especially in the cases of manual editing and geometric deformation or stretching, meanwhile there also exist a number of incorrect feature matches between some totally irrelevant images. This may cause some relevant images to be ranked after the irrelevant ones. To improve the quality of initial search results, various post-processing techniques are proposed to consider additional clues such as the geometric location of local features [4], the extra

* Corresponding author.
E-mail addresses: 198808xc@gmail.com (L. Xie), qitian@cs.utsa.edu (Q. Tian), zhwg@ustc.edu.cn (W. Zhou), dcszb@mail.tsinghua.edu.cn (B. Zhang).

1077-3142/$ - see front matter © 2014 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.cviu.2013.12.011

features from the top-ranked images [5], and the affinity values propagated between images [6]. Although all the methods are verified effective to improve the precision and/or recall of the search results, the handcrafted manners of re-ranking limit them from being generalized and scaled up in the Web environments.

In this paper, we investigate the image search problem from a graph-based perspective, and discover a natural way of re-ranking the initial search results without using handcrafted tricks. First, we illustrate a small graph of near-duplicate images with distractors in Fig. 1. In the image graph, two nodes are linked with each other iff they share no less than 10 common features. Some relevant image pairs are not matched since the local descriptors are quantized into different visual words, whereas the low descriptive power of low-level features also causes the false matches found between some totally irrelevant pairs. Intuitively, given a near-duplicate query image, the missing links between query and the relevant images causes low **recall**, and the false matches between query and the irrelevant images result in low **precision**. To overcome, we claim that image-level matching is more reliable than feature-level matching, and make the following observations which are pivotal to promote the relevant images and filter the irrelevant ones.

- It is possible to access the missing true positives, *i.e.*, relevant images sharing few common features with the query, via constructing the indirect paths through other relevant images.
- For most queries, there are more true positives than false positives in the top-ranked search results. Therefore, it is possible to adopt majority voting or affinity propagation algorithms to filter the false candidates.

Based on the intuitions above, we propose ImageWeb, a novel data structure to capture the image-level context properties. Essentially speaking, ImageWeb is a sparse graph in which each image is represented by a node. There exist an edge from node $\mathbf{I}_a$ to node $\mathbf{I}_b$ if and only if image $\mathbf{I}_b$ appears among the top of the initial search result of image $\mathbf{I}_a$. Since the links in ImageWeb actually imply the recommendation such as "$\mathbf{I}_a$ thinks $\mathbf{I}_b$ is relevant", it is straightforward to adopt the query-dependent link analysis algorithms, say, HITS [7], to re-rank the initial search results by propagating affinities through the links. We verify that, with the efficient discovery of image contexts, it is possible to achieve very accurate search results even without using the handcrafted rules such as spatial verification.

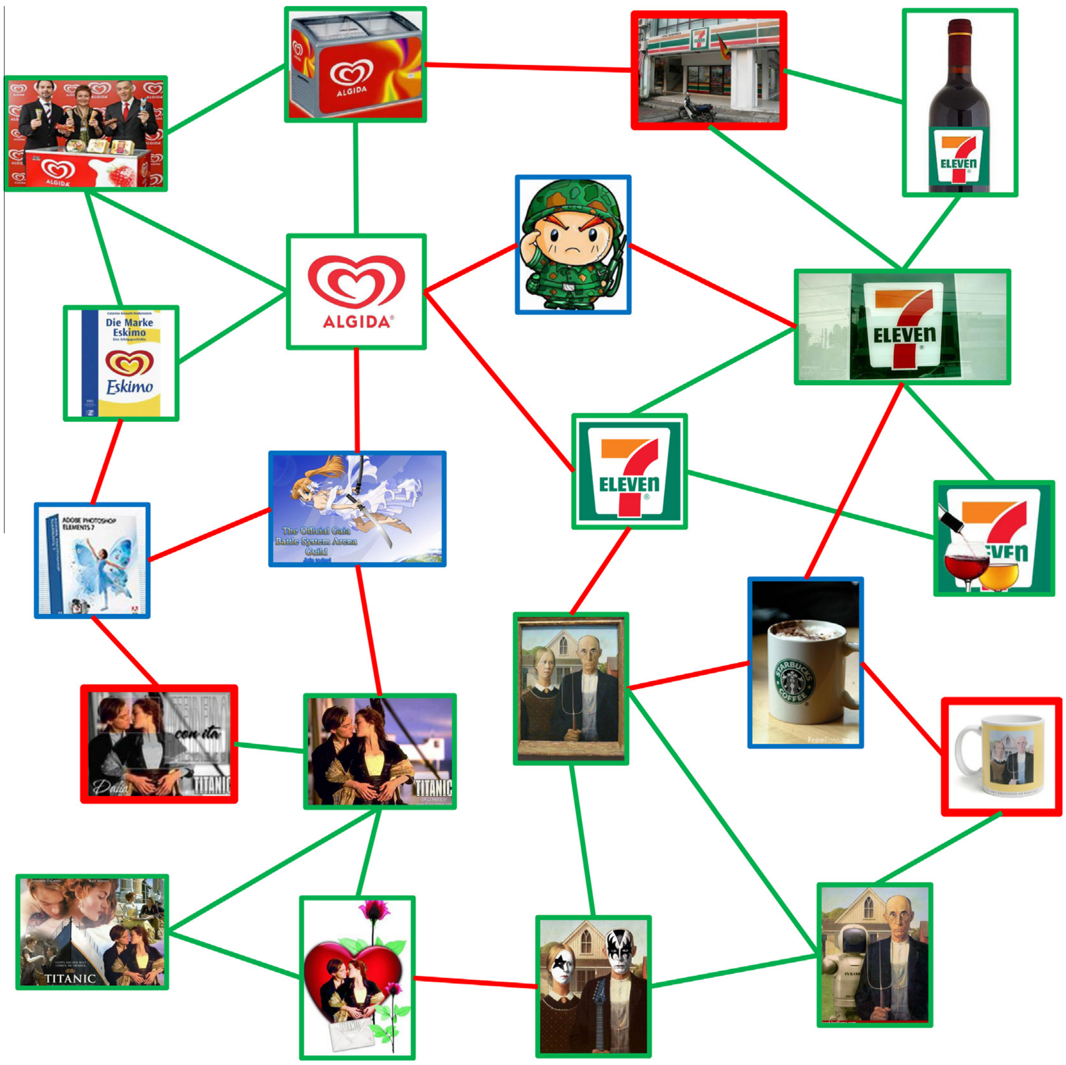

Fig. 1. A toy example of near-duplicate image search with distractors (marked with blue boxes). Images pairs sharing no less than 10 common features are connected with line segments, where green ones and red ones indicate correct and incorrect image matches, respectively. In such a noisy graph, it is difficult for the BoVW-based models to produce high-quality search results, especially on those difficult cases (marked in red boxes). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The major contributions of this work are summarized as follows.

1. We propose an efficient data structure, ImageWeb, to discover the high-level relationships of images. We carefully design the construction, insertion and deletion algorithms to make the data structure efficiency in real applications.
2. We provide a tradeoff strategy to guide the parameter selection in the online searching process. By sacrificing the initial search accuracy which could be compensated in the post-processing, we achieve much better search performance with much lower time complexity compared to the baseline methods.
3. A new near-duplicate dataset with 51 groups of famous car logos are provided. There are many (more than 200) samples in one near-duplicate group which is more similar to the Web environments. Our algorithm works very well in this dataset, demonstrating its good transportability to real-world applications.

The remainder of this paper is organized as follows. First, we formulate a general pipeline for image search in Section 2. In Section 3, we introduce ImageWeb, an efficient data structure to capture the image-level relationships, and use ImageWeb to re-rank initial results in the large-scale image search problems. Section 4 follows to provide a tradeoff strategy for parameter selection in the online search process. After studying the search performance in two challenging databases in Section 5, we draw the conclusions in Section 6.

2. The image search pipeline

In this section, we provide a brief overview of the image search pipeline based on the Bag-of-Visual-Words (BoVW) model and the inverted index structure.

2.1. The Bag-of-Visual-Words model

The Bag-of-Visual-Words (BoVW) model represents each image as a set of visual words. It starts with an image $\mathbf{I} = (a_{ij})_{H \times W}$, where a_{ij} is the **pixel** on position (i, j).

Due to the limited descriptive power of raw image pixels, a number of **regions-of-interest** (ROI) are detected using operators such as DoG [2], MSER [8] or Hessian Affine [9], and the **local descriptors** are calculated on the ROIs to give a basic representation of local patches. Popular descriptors for image retrieval include SIFT [2,10], SURF [11], BRIEF [12] and so on. Each combination of detector and descriptor yields a set of descriptors $\mathcal{D}$:

$$\mathcal{D} = \{(\mathbf{d}_1, \mathbf{l}_1), (\mathbf{d}_2, \mathbf{l}_2), \ldots, (\mathbf{d}_M, \mathbf{l}_M)\}, \tag{1}$$

where $\mathbf{d}_m$ and $\mathbf{l}_m$ denote the **description vector** and the corresponding **location** of the mth descriptor, respectively. M is the total number of descriptors, which could be hundreds or even thousands in common images.

After descriptors have been extracted, they are often quantized into **visual words** to be compact. There are generally two ways of transforming descriptors into visual words. The first method, **Vector Quantization** (VQ), requires training a **codebook** using clustering methods [1]. In image search tasks, K is often rather large, *e.g.*, more than one million, and it could be computationally intractable to compute the accurate (exact) K-Means clustering. Therefore, the hierarchical [3] or approximate [13] versions of K-Means are often adopted for efficiency. After codebook construction, descriptors are quantized into **visual words**. Given a codebook with B codewords, **hard quantization** strategy produces a unique number f_m (from 0 to $B-1$), which is the ID of the nearest (most similar) codeword of descriptor $\mathbf{d}_m$. Oppositely, **soft assignment** maps $\mathbf{d}_m$ onto a subspace spanned by several codewords [14], giving a few different word IDs as well as their weights. In most cases, soft quantization produces more accurate feature matches but also requires more computational resources. We denote the quantization result of $\mathbf{d}_m$ as $\mathbf{f}_m$, which is a sparse, ℓ_1-normalized vector with K dimensions, and each nonzero dimension represents the weight of the corresponding codeword.

The second approach, **Scalar Quantization** (SQ) [15], binarizes each descriptor bin directly without training a codebook. Given a SIFT descriptor $\mathbf{d}_m$ $(D = 128)$:

$$\mathbf{d}_m = (d_{m,1}, d_{m,2}, \ldots, d_{m,D}) \tag{2}$$

a transformation function is defined to quantize $\mathbf{d}_m$ into a **bit vector** (all elements are either 0 or 1) $\mathbf{f}_m$ directly:

$$\mathbf{f}_m = (f_{m,1}, f_{m,2}, \ldots, f_{m,2D}) \tag{3}$$

Here, the **visual word** $\mathbf{f}_m$ is twice the length of **descriptor** $\mathbf{d}_m$, which implies that 2 bits are used to encode each bin of $\mathbf{d}_m$. Following [15], $\mathbf{f}_m$ is calculated in a bitwise manner. For $j = 1, 2, \ldots, D$,

$$(f_{m,j}, f_{m,j+D}) = \begin{cases} (1,1), & \text{if} \quad \hat{d}_m^{\mathrm{H}} < d_{m,j} \\ (1,0), & \text{if} \quad \hat{d}_m^{\mathrm{L}} < d_{m,j} \leqslant \hat{d}_m^{\mathrm{H}} \\ (0,0), & \text{if} \quad d_{m,j} \leqslant \hat{d}_m^{\mathrm{L}} \end{cases} \tag{4}$$

$\hat{d}_m^{\mathrm{H}}$ and $\hat{d}_m^{\mathrm{L}}$ are high and low thresholds, respectively, which are defined by sorting the elements of $\mathbf{d}_m$ in ascending order:

$$\mathbf{d}_m^{\mathrm{S}} = \left(d_{m,1}^{\mathrm{S}}, d_{m,2}^{\mathrm{S}}, \ldots, d_{m,D}^{\mathrm{S}}\right) \tag{5}$$

and using statistics:

$$\hat{d}_m^{\mathrm{L}} = \frac{d_{m,D/2}^{\mathrm{S}} + d_{m,D/2+1}^{\mathrm{S}}}{2} \tag{6}$$

$$\hat{d}_m^{\mathrm{H}} = \frac{d_{m,3D/4}^{\mathrm{S}} + d_{m,3D/4+1}^{\mathrm{S}}}{2}. \tag{7}$$

In the Scalar Quantization framework, the comparison of descriptors in ℓ_2-distance is captured by the Hamming distance between the corresponding $2D$-bit binary vectors.

Of course, more quantization methods such as metric learning [16,17], dimension reduction [18,19] could also be adopted to encode the descriptors. After quantization, a set of visual words are obtained:

$$\mathcal{F} = \{(\mathbf{f}_1, \mathbf{l}_1), (\mathbf{f}_2, \mathbf{l}_2), \ldots, (\mathbf{f}_M, \mathbf{l}_M)\}. \tag{8}$$

Meanwhile, the set of local descriptors are not used in the later steps. Although the computation with visual words are much more effective, the quantization step might discard considerable information, which might cause some irrelevant descriptors projected onto the same visual word. This further weaken the descriptive power of local features.

Besides local descriptors, there are also efforts to represent an image with their global properties. Such holistic features, such as GIST [20] or visual attributes [21], are verified effective to encode the images with relatively smaller number, usually less than a few thousand, of bits. It is acknowledged that global features could serve as a good compensation to local ones, and they are especially useful when retrieving images with close semantics but are not near-duplicate [22–24]. Similar to the classification models combining multiple sources of features [25], local and holistic features have also been integrated in the image search tasks to boost the search performance [26,27]. Since this paper focuses on near-duplicate image search, we do not adopt holistic features in our system.

2.2. The inverted index structure

In large-scale image search, the feature matching problem could be formulated as finding features' approximately nearest neighbors. When the number of features becomes very large, say, over 1 billion, we need an efficient data structure to organize them so that we can search them in a very short time. To address the problem, we leverage from information retrieval [28] to use the inverted index [13,4] as a scalable indexing structure for storing a large corpus of images with their features. In essence, the inverted index is a compact representation of a sparse matrix, whose rows and columns denote features and images, respectively. The offline indexing stage maps each unique codeword onto an entry followed by a list of units, in which we store the ID of the image where the visual word appears, and some other clues for verification, *e.g.*, location of the feature. Moreover, each feature could also be assigned with the IDF score [28] or ℓ_p-IDF score [29], which is used to indicate the feature importance (discriminative power) in the whole corpus.

The ways of indexing visual words in Vector Quantization and Scalar Quantization are also different. Vector Quantization produces a single ID or a group of (weighted) IDs for each descriptor, which could be used as the address of the inverted index entry directly. In contrast, the output of Scalar Quantization is a $2D$-bit binary vector. We take its first t bits as a natural **codeword** to access the entry, and store the remaining $2D - t$ bits as well as the image ID in the indexed unit. It is worth noting that, although the total amount of possible codewords is 2^t, which could be as many as 4G when $t = 32$ [15], the number of valid codewords (with non-empty lists) is much smaller (80M in experiments). Therefore it is possible to allocate an entry for each codeword and use hashing to map the codewords onto index entries.

2.3. Searching process

Given a query image $\mathbf{I}_q$, local descriptors are also extracted and quantized into visual words. Then the searching process finds the corresponding entry of each encoded feature and retrieve the followed lists. Finally, the retrieved images are ranked by their frequencies of occurrence or accumulated IDF scores. The performance of such a naive searching process depends highly on the accuracy of feature matches. However, due to the limited descriptive power of local descriptors and the information loss in the quantization step, the simple counting method often suffers from unsatisfied precision and recall. To improve the quality of search results, various types of post-processing algorithms are used, including **spatial verification**, **query expansion**, and/or **diffusion-based methods**.

It is verified in many works that spatial contexts are useful for filtering false positive results. Therefore, spatial verification [13,5,30] is proposed to boost the **precision** by checking the relative geometric locations of the matched features. For example, [5] considers weak geometric consistency to quickly filter potential false matches, [13] performs global spatial verification based on a variation of RANSAC, and [31,32] adopts a geometric coding scheme by encoding the relative spatial locations within a compact binary code. In [33,34], the authors propose visual phrases as more robust feature groups for filtering false matches.

Query expansion [4,35,36], leveraged from text retrieval, reissues the initial top-ranked results to find valuable features which are not present in the original query. The enriched feature set is helpful to boost the **recall** of search systems. Typical query expansion techniques include transitive closure expansion [4], intra- and inter-expansion [35], automatic failure recovery [36] and so on. In the Scalar Quantization algorithm [15], a spacial approach is adopted for query expansion. The goal is to find the matched visual words (256-bit binary codes) with the Hamming distance no larger than κ. Recall that each visual word in the database is indexed by the first $t = 32$ bits. For each querying feature, the first t bits are also extracted as the querying codeword. The inverted lists with codewords no more than d Hamming distance to the querying codeword are visited, and each of the features in the lists are checked if the total Hamming distance (on 256 bits) to the querying feature are no more than κ. It requires visiting $Q = \sum_{s=0}^{d}\binom{t}{s}$ inverted lists. We name d and κ the **codeword expansion threshold** and **Hamming threshold**, respectively. Note that a larger expansion threshold often results in heavier enumeration (larger Q) and therefore higher time complexity.

In image search, it is also straightforward to propagate affinities or beliefs via the connection of objects. Based on global image matching, it is feasible to establish an image network [6], and calculate the affinity scores of image candidates in the initial search results. For example, [37,6,38] explore the use of content-based features to improve the quality of text-based image search results, and [39] mines both context and content links in social media networks to discover the underlying latent semantic space. In [27], the relationship between images are presented using a graph-based data structure, which is also exploited in [40] for exploring image collections based on global topological analysis.

The proposed ImageWeb algorithm is an efficient implementation of diffusion methods. Different from the related work, VisualRank [6], which adopts PageRank to improve text-based image retrieval, we leverage the query-dependent HITS algorithm into the content-based image retrieval task with clear intuitions. We shall also highlight the superior performance and fast online searching speed as our contributions.

3. The ImageWeb

This section illustrates ImageWeb for re-ranking images to improve the initial search quality. We shall first introduce the intuition originating from the document retrieval community, and then present the construction, updating and searching algorithms of ImageWeb. Finally, we discuss the scalability of ImageWeb by analyzing its time and memory consumptions.

3.1. Document search and link analysis

Document search engines play an important role on the World Wide Web (WWW). A search engine typically consists of three major stages. First, webpages are found by a **Web crawler**, an automated Web browser which follows every link on the websites. The retrieved webpages are then stored in an **index database** for later use. When a user types in the querying keywords, the engine looks up the index and retrieves a list of best-matched web pages according to some criteria. While there may be millions of webpages that include the querying keywords, some pages may be more relevant, popular, or authoritative than others. Most search engines employ link analysis to rank the retrieved webpages and return the best ones to the users.

To measure the quality of webpages, many link analysis algorithms explore the internal structure of the Web, and use random walk, a mathematical formulation to calculate the affinity values [41] of the pages. The most popular link analysis algorithms include PageRank [42] and HITS [7]. Both of them assume that high quality websites are likely to receive more links from other websites. and the quality of a webpage could be roughly estimated by accumulating affinities from the pages linking to it. Both PageRank and HITS could be computed either intuitively with iteration methods, or mathematically using eigenvectors.

PageRank [42] assigns a numerical weight $w \in (0, 1)$ to each element of the Web. which is a probability representing the likelihood that a person randomly clicking on links will arrive at the particular page. In each iteration, each node distribute its weight through its outgoing links and re-collects weight from the incoming links. Since we set equal weights on each page at the beginning of the algorithm, the final results of PageRank is a reflection of the Web's intrinsic structure, and therefore independent to any specified queries.

As a precursor of PageRank, HITS [7] algorithm, or known as Hubs and Authorities, also assigns numerical weights $h, a \in (0, 1)$ named hubs and authorities to each webpage, which stems from a particular insight into the creation of webpages when the Web was originally forming. In each iteration, the authority values are first updated using the hub values, and the hub values are then updated using the new version of authority values. Different from PageRank, HITS starts with a given query, and the hub and authority values of the query are set to 1 while others are 0. Therefore, HITS is a query-dependent algorithm.

3.2. ImageWeb: definition and construction

The key assumption of the link analysis algorithms in document retrieval is that one webpage $\mathbf{P}_a$ contains a link to another one $\mathbf{P}_b$ if $\mathbf{P}_a$ suggests that $\mathbf{P}_b$ is of high quality. In order to leverage those algorithms into the image search problems, we organize the images in the database with a graph model, in which each image is represented by a node, and there exists a link from image $\mathbf{I}_a$ to $\mathbf{I}_b$ if $\mathbf{I}_b$ is ranked among the top in the (initial) search results of $\mathbf{I}_a$. The same assumption also holds in the constructed network of images. One image $\mathbf{I}_a$ contains a link to another one $\mathbf{I}_b$ implies that $\mathbf{I}_b$ is more likely to be relevant to $\mathbf{I}_a$.

Formulating the basic idea above yields the definition of ImageWeb, a data structure for organizing image-level relationships. Suppose we have a large-scale image database $\mathcal{I} = \{\mathbf{I}_1, \mathbf{I}_2, \ldots, \mathbf{I}_N\}$, where N is the total number of images which could be over one million in real applications. **ImageWeb** is a directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ is the set of nodes, and v_n is the corresponding node of image $\mathbf{I}_n$ for $n = 1, 2, \ldots, N$. $\mathcal{E}$ is the set of edges connecting between nodes, and there is an directed edge from v_{n_1} to v_{n_2} iff $\mathbf{I}_{n_2}$ appears among the top-K (initial) search results of $\mathbf{I}_{n_1}$. Here K, an integer much smaller than N, is named the **breadth** of the ImageWeb $\mathcal{G}$. There is also a weight assigned to each edge of the ImageWeb, indicating the importance of the edge. In any time, the outgoing links of any node are ranked according to their weights, and the weights of the node are normalized so that their sum equals to 1. The algorithm of constructing ImageWeb is illustrated in Fig. 2.

In real applications, the image database might change with time. Newly retrieved images might be added, while some outdated or illegal images should be removed from the database. Therefore, it is necessary for the ImageWeb to support efficient insertion and deletion operations. The main difficulties of image insertion and deletion arise from updating the outgoing links of images that are already in the database before insertion or preserved after deletion. Re-calculating the outgoing links for all the images is a natural solution, but it could be computationally intractable if we re-construct the whole ImageWeb after each modification of image database. Here, we adopt the approximated updating algorithm, in which we only check and update those images that are linked by the inserted/deleted images. The complete re-calculation of one image's outgoing links is only called when necessary, *i.e.*, the number of outgoing links is less than a threshold, say, $0.8\,K$. Since the approximated updating algorithm might cause the ImageWeb more and more inaccurate, we can re-construct the whole ImageWeb after a relatively long period of time. The insertion and deletion algorithms are illustrated in Figs. 3 and 4, respectively.

3.3. ImageWeb: link analysis

Fig. 5 illustrates two nodes in a toy ImageWeb ($K = 10$) constructed on 1104 images. We present an example for the easy and difficult queries, respectively. It is worth noting that K is usually a very small number relative to the size of database, therefore we are actually not storing all the true positives to each query in the ImageWeb. The ImageWeb only provides a pre-calculated network structure for improving the quality of online search results.

The link analysis algorithm on the ImageWeb is aimed at filtering the top-ranked false positives for each query. Here, we benefit from the fact that all the true positive images contain the same near-duplicate concept, and it is more probable to find effective feature matches between them, *i.e.*, they are more likely to appear among others' outgoing links (top-ranked candidates). Even when we consider the relatively more difficult case in Fig. 5, in which more false positives (6 out of 10) are found in the top-ranked results, we can see that 6 false samples come from 6 different near-duplicate groups, therefore they are not likely to be connected with each other very closely.

The above observations inspire us to adopt the affinity propagation algorithms for image re-ranking. Given a query image, we extract all its visual words to look up the inverted index, and obtain a N-dimensional vector representing the number of feature matches between the query and each candidates. Normalizing the vector yields a probability distribution over all the images $\mathbf{w}_0 = (w_{0,1}, w_{0,2}, \ldots, w_{0,N})$ satisfying $\sum_n w_{0,n} = 1$. We take this distribution as the initial hub values of all the images, and adopt

```
Algorithm 1: Building ImageWeb.

 1. Input.
    A large image database $\mathcal{I} = \{\mathbf{I}_1, \mathbf{I}_2, \ldots, \mathbf{I}_N\}$, ImageWeb breadth $K$.
 2. Indexing and initialization.
    Index all the features in $\mathcal{I}$ into an inverted file. Initialize $\mathcal{V} = (v_1, v_2, \ldots, v_N)$. Here $v_n$ is the node
    corresponding to image $\mathbf{I}_n$, for $n = 1, 2, \ldots, N$.
 3. Link construction.
    For each $v_n$, take image $\mathbf{I}_n$ as query and search in the inverted index. Add an directed edge from $v_n$
    to each node corresponding to the $K$ top-ranked images in search results. The weight of an edge is
    the count of matched features between the query and the candidate.
 4. Normalization.
    The weights of a query image is normalized so that they sum to 1.
 5. Output.
    ImageWeb $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$.
```

Fig. 2. The general steps of building ImageWeb.

```
Algorithm 2: Inserting images into ImageWeb.
1. Input.
   Original ImageWeb $\mathcal{G}$, a new database $\mathcal{I}^{\mathrm{i}} = \{\mathbf{I}^{\mathrm{i}}_1, \mathbf{I}^{\mathrm{i}}_2, \ldots, \mathbf{I}^{\mathrm{i}}_{N_{\mathrm{i}}}\}$.
2. Indexing and initialization.
   Append all the features in $\mathcal{I}^{\mathrm{i}}$ into the original inverted file (before insertion operation). Initialize $\mathcal{V}^{\mathrm{i}} = (v^{\mathrm{i}}_1, v^{\mathrm{i}}_2, \ldots, v^{\mathrm{i}}_{N_{\mathrm{i}}})$. Here $v^{\mathrm{i}}_n$ is the node assigned to image $\mathbf{I}^{\mathrm{i}}_n$, for $n = 1, 2, \ldots, N_{\mathrm{i}}$.
3. Update links of $\mathcal{I}^{\mathrm{i}}$.
   For each $v^{\mathrm{i}}_n$, perform an initial search on query $\mathbf{I}^{\mathrm{i}}_n$ to find its top-$K$ candidates and build the outgoing links from $v^{\mathrm{i}}_n$.
4. Updating outgoing links of $\mathcal{I}$.
   For each $v^{\mathrm{i}}_n$, check each of the $K$ top-ranked search results from $v^{\mathrm{i}}_n$, say, $v'_k$, to see if the matching count between $v'_k$ and $v^{\mathrm{i}}_n$ is larger than $v'_k$'s $K$-th (last) outgoing candidate. If so, insert $v^{\mathrm{i}}_n$ into $v'_k$'s outgoing links at a proper position, and remove the last candidate.
5. Normalization.
   The weights of a query image is normalized so that they sum to 1.
6. Output.
   Updated ImageWeb $\mathcal{G}'$.
```

Fig. 3. The general steps of inserting images into ImageWeb.

```
Algorithm 3: Deleting images from ImageWeb.
1. Input.
   Original ImageWeb $\mathcal{G}$, deleting set $\mathcal{I}^{\mathrm{d}} = \{\mathbf{I}^{\mathrm{d}}_1, \mathbf{I}^{\mathrm{d}}_2, \ldots, \mathbf{I}^{\mathrm{d}}_{N_{\mathrm{d}}}\}$.
2. Indexing and initialization.
   Remove all the features in $\mathcal{I}^{\mathrm{d}}$ from the original inverted file (before deletion operation). Delete $\mathcal{V}^{\mathrm{d}} = (v^{\mathrm{d}}_1, v^{\mathrm{d}}_2, \ldots, v^{\mathrm{d}}_{N_{\mathrm{d}}})$. Here $v^{\mathrm{d}}_n$ is the node assigned to image $\mathbf{I}^{\mathrm{d}}_n$, for $n = 1, 2, \ldots, N_{\mathrm{d}}$.
3. Updating outgoing links of $\mathcal{I}$.
   For each $v^{\mathrm{d}}_n$, check each of the $K$ top-ranked search results from $v^{\mathrm{d}}_n$, say, $v'_k$, to see if $v^{\mathrm{d}}_n$ is listed in $v'_k$'s top-$K$ candidates. If so, remove $v^{\mathrm{d}}_n$ from $v'_k$'s candidates.
4. Checking.
   For each preserved image, check how many of its top-$K$ candidates are preserved. If the number is less than a fixed threshold, say, $0.8K$, perform an initial search to re-construct its outgoing links.
5. Normalization.
   The weights of a query image is normalized so that they sum to 1.
6. Output.
   Updated ImageWeb $\mathcal{G}'$.
```

Fig. 4. The general steps of deleting images from ImageWeb.

the HITS algorithm [7] on the ImageWeb $\mathcal{G}$ to update authority and hub values of all the images iteratively for R rounds. R is named the **depth** of link analysis, which means that after R iterations, we could retrieve all the candidates that the query-candidate shortest path is not longer than R in the ImageWeb. We do not always iterate the weighting till convergence for the consideration of time consumption. The algorithm is formulated in Fig. 6.

3.4. Time and memory complexity

In this part, we focus on the time and memory consumptions caused by the ImageWeb algorithms. We ignore the time used at descriptor extraction, quantization and indexing stages, and the memory cost for indexing millions of images into and inverted file structure.

The time overhead of the ImageWeb algorithms consist of two parts, offline construction and online searching. The main part of offline time is used in the initial searching process, and is highly related to the BoVW model adopted to construct the ImageWeb. In practise, we use the Scalar Quantization [15] to generate the initial results. Suppose there are N images in the database, each image contains M features, and each feature is expanded Q times in initial search process, then each image requires $O(MQ\alpha)$ time to access the inverted index, and $O(N\log(N))$ time for sorting the candidates. Here, α is the average number of features indexed after each entry. In real applications, we have $MQ < N$ and $\alpha \sim \log(NM) < 2\log(N)$, therefore the time complexity for constructing the ImageWeb is $O(N^2\log(N))$, Moreover, it requires $O(N\log(N))$ time to insert an image into the ImageWeb or delete an image from the ImageWeb (ignoring the sparsely called re-search operations in image deletion). In our experiments (see Section 5), N is larger than 10^6, and we use parallelization methods to accelerate the construction process. Considering the operation is required only once, the time complexity is affordable.

The online stage requires $O(N\log(N))$ operations for initial searching, and an R-round HITS algorithm consisting of $2NKR$ floating number calculations. With different settings of parameters, *i.e.*, codeword expansion threshold d, Hamming threshold κ (see Section 2.2), ImageWeb breadth K and depth R, the time cost of online stage could vary from tens of milliseconds to tens of seconds. It is a challenge to accelerate the algorithm without harming the search accuracy. We will cope with this problem in Section 4.

The memory cost of the ImageWeb comes from the storage of each image's K top-ranked candidates in the ImageWeb. It costs 8 bytes to store one candidate (4 bytes for image ID and 4 bytes for score). We set $K = 20$ for the best performance of searching process (see Section 4.2), therefore the total memory cost on a single image is 160 bytes. In total, we need 160 Megabytes to store the ImageWeb with one million images, which is much smaller than 4 Gigabytes used for loading the inverted index of one millions images.

4. The tradeoff strategy for parameter selection

This section is aimed at finding a proper set of parameters for the online searching process. For this, we adopt two intuitive tradeoff strategies to guide the parameter selection, one between precision and recall, and the other between time complexity and search accuracy. Finally, accurate search results are achieved with very low time consumptions.

We start with introducing the basic settings used in our experiments.

4.1. The experimental settings

We use Scalar Quantization (SQ) [15] as the baseline system. Based on the initial search results provided by SQ, we construct ImageWeb for post-processing. To make fair comparison, we keep the same settings as the baselines.

- *Descriptor extraction.* We use the SIFT descriptors [2] calculated on the Regions of Interest detected by DoG operators [2]. All the images are greyscale, and resized so that the larger axis size is 300.
- *Descriptor quantization.* We use Scalar Quantization (SQ) [15] formulation (4) to encode each 128-D SIFT descriptor into a 256-bit binary code.

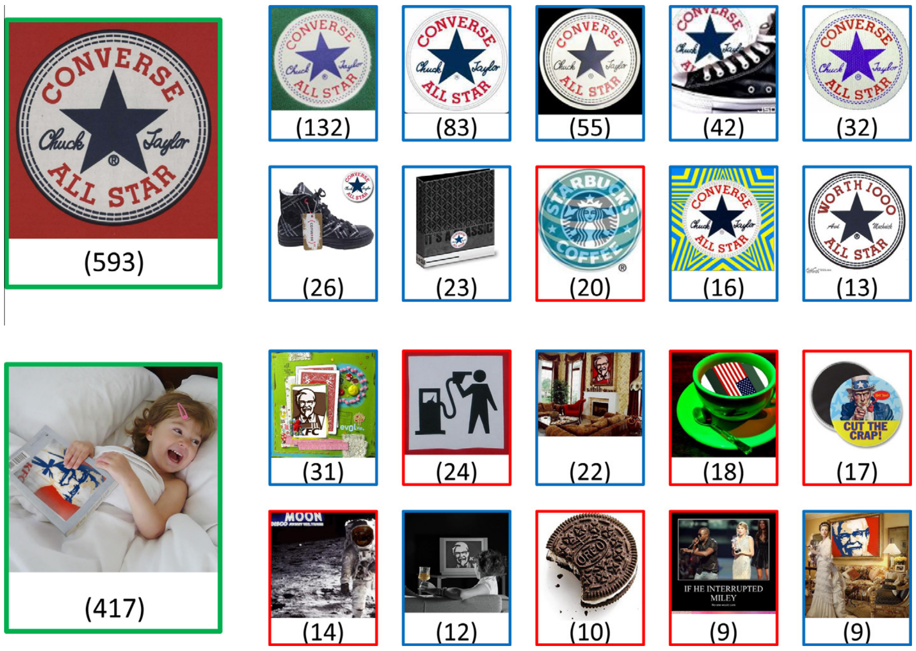

Fig. 5. Two nodes with outgoing links ($K = 10$) of a toy example of ImageWeb ($N = 1104$). Images on the right side are ranked according to the number of feature matches (numbers in parenthesis) to the query image. The false positives in the outgoing links are marked with red boxes. The above node shows an easy query, in which only one false positive is found among the top-10 candidates, while the bottom one is relatively more difficult, in which six false positives are ranked among top-10. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

```
Algorithm 4: Searching on ImageWeb.
  1. Input.
     ImageWeb $\mathcal{G}$, initial weighting (hubs) $\mathbf{w}_0$, ImageWeb depth $R$.
  2. Calculating authorities ($r+1$-st round).
     For $n = 1, 2, \ldots, N$, set $w'_{r+1,n} = 0$, and for each of $v_n$'s incoming neighbor $v_i$, set $w'_{r+1,n} \leftarrow w'_{r+1,n} + w_{r,i}$. Normalize $\mathbf{w}'_{r+1}$ as the authorities of all images.
  3. Calculating hubs ($r+1$-st round).
     For $n = 1, 2, \ldots, N$, set $w_{r+1,n} = 0$, and for each of $v_n$'s outgoing neighbor $v_o$, set $w_{r+1,n} \leftarrow w_{r+1,n} + w'_{r+1,o}$. Normalize $\mathbf{w}_{r+1}$ as the hubs of all images.
  4. Iteration.
     Loop the above steps for $R$ times.
  5. Output.
     $\mathbf{w}_R$, the final weighting for sorting.
```

Fig. 6. General steps of HITS calculation in ImageWeb.

- *Indexing.* The first 32 out of 256 bits of each visual word are taken as the indexing address. Image ID as well as the remaining 224-bit binary codes are stored in the inverted index. We remove the features which appear in more than $N^{1/3}$ images where N is the number of images in the whole corpus.
- *Online searching.* We follow the basic searching process of Scalar Quantization [15] to obtain the initial scores. The HITS algorithm is then performed on a pre-constructed ImageWeb. The impact of searching parameters will be discussed thoroughly in this section.
- *Accuracy evaluation.* We use the mean average precision (mAP) to evaluate the accuracies of all methods.

There are several adjustable parameters that affect the online searching stage, including the **codeword expansion threshold** d and the **Hamming threshold** κ in the initial searching process (Section 2.3), and the ImageWeb **breadth** K and **depth** R (see Section 3). The different sets of parameters could result in contrasting search accuracies and time costs, and our goal is to find a proper set of parameters producing accurate search results very efficiently. It is obviously very difficult to enumerate all the possible parameter combinations and choose the best one. Therefore, we adopt an intuitive tradeoff strategy to help us find a proper set of parameters.

Throughout this section, we report the mAP values and time costs on the DupImage dataset [31] with one million distractors. The detailed information of this dataset could be found in Section 5.1.

4.2. Tradeoff between precision and recall

First we consider the setting of **breadth** K and **depth** R in the ImageWeb. A larger K causes more top-ranked candidates is stored in the ImageWeb structure, and a larger R implies that affinity values are propagated through more iteration rounds. Intuitively, the increase of K and R helps to improve the recall and precision, respectively. As shown in [43,5], there always exists a precision-recall tradeoff in both document and image search systems. For example, decreasing the stopword threshold, *i.e.*, more visual words are stopped, produces more accurate feature matching to improve the search precision, but also rejects some correct matches and therefore degrades the recall.

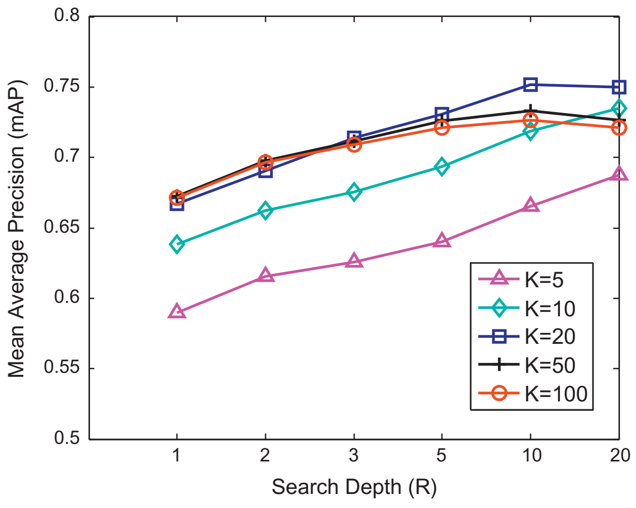

Fig. 7. Performance comparison with respect to different settings of breadth and depth of the HITS algorithm. The mAP values are recorded for each combination of depth $R = 1, 2, 3, 5, 10, 20$ and breadth $K = 5, 10, 20, 50, 100$. Experiments are performed on the DupImage dataset (see Section 5.1) mixed with one million distractors.

To be clear, we test different combinations of K and R, and plot the mAP values under different settings in Fig. 7. It is easy to observe that the mAP value does not always increase when the breadth and depth go up, although large breadth and depth help to improve the recall and precision of search results, respectively. In fact, if K becomes too large, a considerable number of false positives would be introduced into the ImageWeb, which would harm the precision of link analysis algorithm. On the other hand, an improper value of R could also cause the search accuracy drop. Therefore, we can conclude that individually maximizing precision or recall does not provide the best overall performance. The highest mAP value is achieved using the compromised parameters, *i.e.*, $K = 20$ and $R = 10$.

4.3. Tradeoff between initial search and post processing

Next we configure the setting of **codeword expansion threshold** d and **Hamming threshold** κ in initial searching. We report the search accuracy and query time obtained with different sets of parameters in our algorithm, where the time percentages used in initial search and post-processing are recorded, respectively. We list a part of the data in Table 1. When $K = R = 0$ (no post-processing), the search accuracy does increase significantly with the codeword expansion threshold d, since a larger expansion threshold helps to capture more possibly matched features in the expanded searching. The major shortcoming of a large d value is the considerable time use at the online searching stage, *i.e.*, we need about 500 ms to process a query when $d = 2$, and even more than 4 s when $d = 3$. Fortunately, the ImageWeb is verified as a good compensation to the inaccurate initial searching process. With ImageWeb re-ranking algorithms, we can obtain excellent performance even based on very poor initial results (see Table 1, from about 0.15 to higher than 0.75). Moreover, ImageWeb bridges the accuracy gap between small and large d values. Under proper setting of breadth $R = 10$ and depth $K = 20$, the mAP difference between $d = 0$ and $d = 3$ decreases from about 0.40 to no more than 0.01. Considering that $d = 0$ only requires 50 ms for initial searching, which is merely 1% of using $d = 3$, it is quite instructive to adopt it to shorten the time of online searching.

We provide an explanation to the parameter selection using the well-known marginal utility. Suppose we have fixed the total time

Table 1
We list the parameters, searching time and accuracy (mAP) to illustrate the tradeoff between initial search and post-processing. d, κ, K and R are **codeword expansion threshold**, **Hamming threshold**, ImageWeb **breadth** and **depth**, respectively. t_1, t_2 and T are the time (in milliseconds) used in initial search, post-processing and the whole search process.

d	κ	K	R	t_1	t_2	t_2/T (%)	mAP
0	16	0	0	50	0	0.00	0.143
0	16	10	5	50	16	24.24	0.693
0	16	10	10	50	31	38.27	0.728
0	16	20	5	50	30	37.50	0.731
0	16	20	10	50	60	54.55	0.752
1	16	0	0	80	0	0.00	0.428
1	16	10	5	80	16	16.67	0.724
1	16	10	10	80	31	27.93	0.742
1	16	20	5	80	30	27.27	0.737
1	16	20	10	80	60	42.86	0.755
2	16	0	0	460	0	0.00	0.515
2	16	10	5	460	16	3.36	0.733
2	16	10	10	460	31	6.31	0.753
2	16	20	5	460	30	6.12	0.749
2	16	20	10	460	60	11.54	0.757
3	16	0	0	4240	0	0.00	0.537
3	16	10	5	4240	16	0.38	0.741
3	16	10	10	4240	31	0.73	0.756
3	16	20	5	4240	30	0.70	0.756
3	16	20	10	4240	60	1.40	0.762
0	24	0	0	54	0	0.00	0.161
0	24	10	5	54	16	22.86	0.699
0	24	10	10	54	31	36.47	0.726
0	24	20	5	54	30	35.71	0.730
0	24	20	10	54	60	52.63	0.756
1	24	0	0	86	0	0.00	0.451
1	24	10	5	86	16	15.69	0.726
1	24	10	10	86	31	26.50	0.745
1	24	20	5	86	30	25.86	0.744
1	24	20	10	86	60	41.10	0.758
2	24	0	0	477	0	0.00	0.542
2	24	10	5	477	16	3.25	0.731
2	24	10	10	477	31	6.10	0.756
2	24	20	5	477	30	5.92	0.752
2	24	20	10	477	60	11.17	0.761
3	24	0	0	4320	0	0.00	0.563
3	24	10	5	4320	16	0.37	0.744
3	24	10	10	4320	31	0.71	0.755
3	24	20	5	4320	30	0.69	0.759
3	24	20	10	4320	60	1.37	0.764

cost in the online searching process (including initial searching and post-processing), then it is unadvisable to spend too large fraction in either initial or post processing stage. When we choose $d = 0, \kappa = 16$ and $K = 20, R = 10$, the initial and post processing stages would require 50 ms and 60 ms for each query, respectively. The total searching time, 110 ms, is even much shorter than the baseline system (480 ms). We shall inherit these parameters in the later experiments.

5. Experimental results

We compare our algorithm with the following popular methods:

1. HVVT [3] is the original BoVW-based framework with **Hierarchical Visual Vocabulary Tree**. We train a codebook with one million leaf codewords (6 layers, at most 10 branches at each node).
2. HE [5] uses **Hamming Embedding** to filter the candidate features which are quantized to the same codeword but have large Hamming distances to the query feature. The threshold for Hamming distance is selected as 20 for the best performance.

Fig. 8. Sample images from the DupImage dataset.

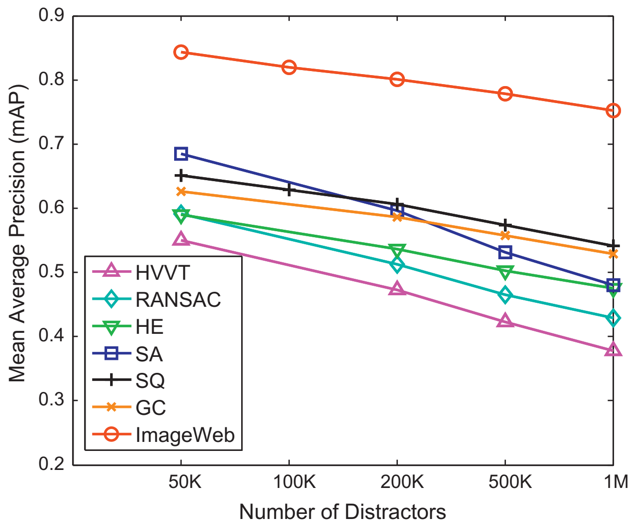

Fig. 9. Performance comparison on the DupImage dataset with different numbers of distractors.

3. RANSAC [13] performs image re-ranking after initial search via geometric verification, which is based on the estimation of an affine transformation by a variant of **Random Sampling Consensus**.
4. SA [14] exploits **Soft Assignment** to identify a local descriptor with a representation of nearby codewords. For the accuracy-efficiency tradeoff, we set the error bound in the k–d tree as 5.
5. SQ [15] gives an efficient image search flowchart based on codebook-free **Scalar Quantization** (4). Next, the features are indexed by their first 32 out of 256 quantized bits. Following [15], we select the codeword expansion threshold $d = 2$ and Hamming threshold $\kappa = 24$.
6. GC [31] adopts a **geometric coding** scheme for local feature match verification by encoding the relative spatial locations among features within an image and discovering false feature matches between images.

5.1. The DupImage dataset

The DupImage dataset [31] contains 1104 images from 33 groups, including logos (*e.g.*, ‘Starbucks’, ‘KFC’), pictures (*e.g.*, ‘Sailor Kiss’), paints (*e.g.*, ‘Mona-Lisa’) and other near-duplicate signs. Samples images are shown in Fig. 8. We mix the dataset with one million distractors crawled from the Internet. To evaluate the performance with respect to the number of distractors, we construct 4 smaller subsets of distractors by random sampling. The sizes of the subsets are 50 K, 100 K, 200 K and 500 K, respectively. We evaluate our algorithm as well as 6 baseline systems on the 5 different sizes of datasets, and record the mAP values and the time costs per query for comparison.

The mAP values are plotted in Fig. 9. It is observed that our approach significantly outperforms all the other methods on all the reported sizes of datasets. With one million distractors, the best candidate system, SQ [15], gives a 0.542 mAP. Our approach hits 0.752, giving a relatively 38.7% improvement over SQ.

5.2. The CarLogo-51 dataset

We manually collect and label the CarLogo-51 dataset, which contains 51 categories of car logos. [1] There are at least 200 images in each category and the total number of images is 11903. Samples images are shown in Fig. 10. We also mix the basic set with 10 K, 20 K, 50 K, 100 K, 200 K, 500 K and 1 M distractors, respectively.

The mAP values are plotted in Fig. 11. Again our algorithm beats all the competing algorithms significantly, and enjoys a surprisingly 102% relative improvement of mAP value (0.426) over SQ (0.211), the best candidate search system, with one million distractors.

5.3. Time cost

The offline stages of our algorithm are the same as Scalar Quantization [15], which takes roughly 200 ms for processing one image including descriptor extraction, quantization and indexing. Here we mainly report the time costs in two stages: ImageWeb con-

[1] This dataset is publicly available online. Type ’CarLogo-51 dataset’ for searching.

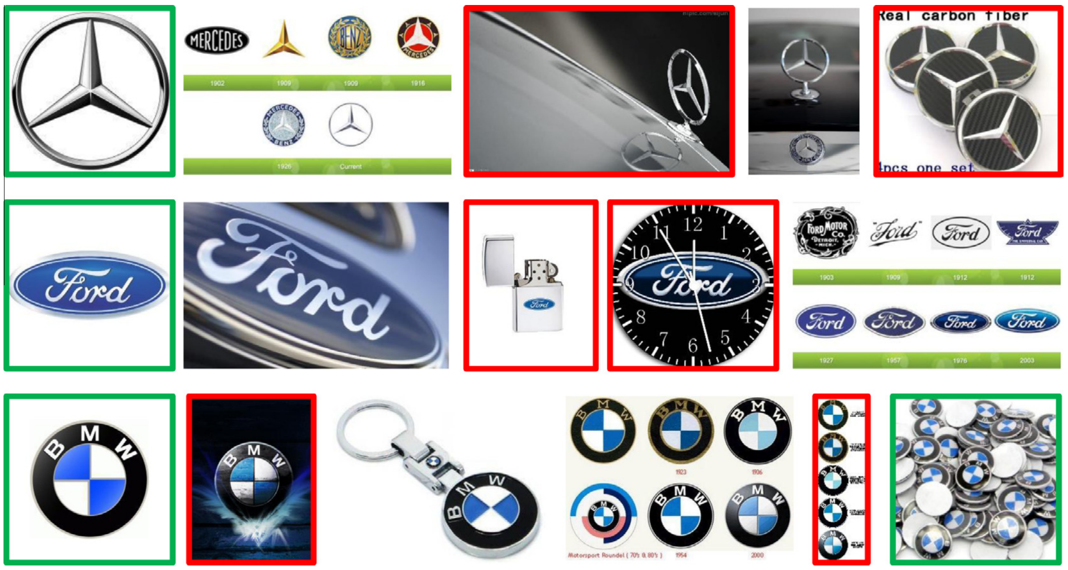

Fig. 10. Sample images from the CarLogo-51 dataset. Green boxes indicate standard logos, and red ones are difficult cases. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

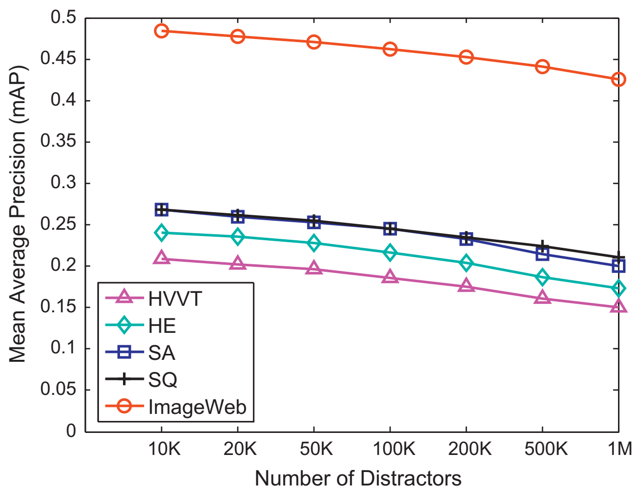

Fig. 11. Performance comparison on the CarLogo-51 dataset with different numbers of distractors.

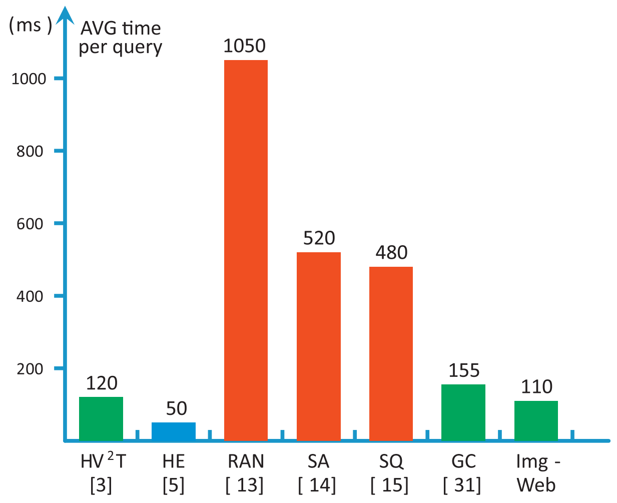

Fig. 13. Average querying time (in milliseconds) of different methods on the database with one million images.

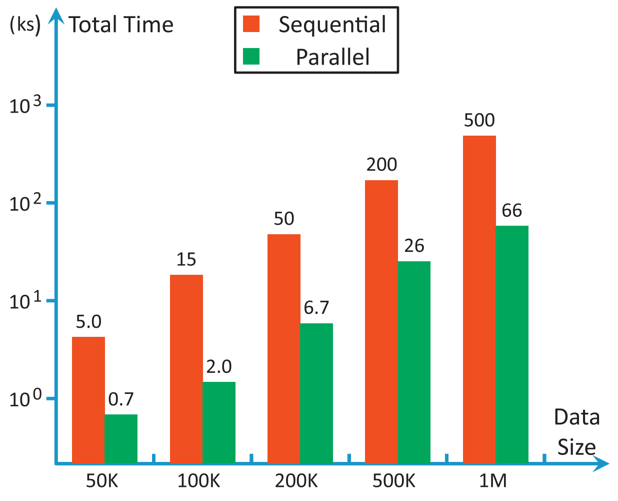

Fig. 12. Time cost (in kiloseconds) of constructing ImageWeb with respect to the size of image database.

struction and online searching. The theoretical analysis of time complexity could be found in Section 3.4.

We plot the required time to construct ImageWeb with respect to the number of images in Fig. 12. The construction algorithm is highly parallelizable. Considering the offline construction of ImageWeb is performed only once, the time cost (less than one day for one million images) is affordable.

The average time cost on each query of different approaches are plotted in Fig. 13. Our algorithm needs only 110 ms to process one query, which is faster than all the comparison algorithms except [5], and even more than 4 times faster than the baseline system, Scalar Quantization [15]. In fact, our algorithm benefits from the much looser initial search parameters, *i.e.*, we use $d = 0$ and $\kappa = 16$ while [15] uses $d = 2$ and $\kappa = 24$, which immediately cuts the initial search time down to one tenth (from 480 ms to about 50 ms). Considering the supreme search performance we obtain, the low time complexity is really surprising and exciting.

6. Conclusions

In this paper, we investigate the near-duplicate Web image search problem, and propose a novel solution for refining the initial search results. We observe that the shortcoming of BoVW-based image search framework arises from the limited descriptive power of local features, and claim that the image-level matches are more reliable than the feature-level matches. From a graph-based perspective, we propose ImageWeb, an efficient data structure to capture images' context properties, and adopt HITS, a query-dependent re-ranking algorithm, to bring order into the ImageWeb. An intuitive tradeoff strategy is also introduced to guide the parameter selection at the online searching stage. Experimental results on the large-scale image search datasets reveal that our algorithm achieves significant accuracy improvement with even much lower time complexity compared to the baseline system. Moreover, our algorithm is highly scalable, as the time and memory overheads are both linear to the size of the image corpus.

Acknowledgements

This work was supported by the National Basic Research Program (973 Program) of China (Grant Nos. 2013CB329403 and 2012CB316301), the National Natural Science Foundation of China (Grant No. 61332007, 61273023 and 91120011), the Beijing Natural Science Foundation (Grant No. 4132046), and the Tsinghua University Initiative Scientific Research Program (Grant No. 20121088071). This work was also supported in part to Dr. Qi Tian by ARO grant W911NF-12-1-0057, Faculty Research Awards by NEC Laboratories of America, and 2012 UTSA START-R Research Award respectively. This work was also supported in part by National Science Foundation of China (NSFC) 61128007. This work was also supported in part to Dr. Zhou by the Fundamental Research Funds for the Central Universities and the start-up funding from the University of Science and Technology of China.

References

[1] J. Sivic, A. Zisserman, Video Google: a text retrieval approach to object matching in videos, in: International Conference on Computer Vision, 2003.
[2] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vision (2004).
[3] D. Nister, H. Stewenius, Scalable recognition with a vocabulary tree, in: Computer Vision and Pattern Recognition, 2006.
[4] O. Chum, J. Philbin, J. Sivic, M. Isard, A. Zisserman, Total recall: automatic query expansion with a generative feature model for object retrieval, in: International Conference on Computer Vision, 2007.
[5] H. Jegou, M. Douze, C. Schmid, Hamming embedding and weak geometric consistency for large scale image search, in: European Conference on Computer Vision, 2008.
[6] Y. Jing, S. Baluja, VisualRank: applying pagerank to large-scale image search, in: IEEE Transactions on Pattern Analysis and Machine Intelligence, 2008.
[7] J.M. Kleinberg, Authoritative sources in a hyperlinked environment, J. ACM (1999).
[8] J. Matas, O. Chum, M. Urban, T. Pajdla, Robust Wide-Baseline Stereo from Maximally Stable Extremal Regions, in: Image and Vision Computing, 2004.
[9] K. Mikolajczyk, C. Schmid, Scale & affine invariant interest point detectors, Int. J. Comput. Vision (2004).
[10] R. Arandjelovic, A. Zisserman, Three things everyone should know to improve object retrieval, in: Computer Vision and Pattern Recognition.
[11] H. Bay, T. Tuytelaars, L. Van Gool, SURF: speeded up robust features, in: European Conference on Computer Vision, 2006.
[12] M. Calonder, V. Lepetit, C. Strecha, P. Fua, BRIEF: binary robust independent elementary features, in: European Conference on Computer Vision, 2010.
[13] J. Philbin, O. Chum, M. Isard, J. Sivic, A. Zisserman, Object retrieval with large vocabularies and fast spatial matching, in: Computer Vision and Pattern Recognition, 2007.
[14] J. Philbin, O. Chum, M. Isard, J. Sivic, A. Zisserman, Lost in quantization: improving particular object retrieval in large scale image databases, in: Computer Vision and Pattern Recognition, 2008.
[15] W. Zhou, Y. Lu, H. Li, Q. Tian, Scalar quantization for large scale image search, in: ACM Multimedia, 2012.
[16] S.C. Hoi, W. Liu, M.R. Lyu, W.-Y. Ma, Learning distance metrics with contextual constraints for image retrieval, in: Computer Vision and Pattern Recognition, 2006.
[17] B. Geng, D. Tao, C. Xu, DAML: domain adaptation metric learning, in: IEEE Transactions on Image Processing, 2011.
[18] N. Guan, D. Tao, Z. Luo, J. Shawe-Taylor, MahNMF: manhattan non-negative matrix factorization, in: arXiv preprint, 2012.
[19] T. Zhou, D. Tao, Double shrinking for sparse dimension reduction, in: IEEE Transactions on Image Processing, 2013.
[20] A. Oliva, A. Torralba, Modeling the shape of the scene: a holistic representation of the spatial envelope, Int. J. Comput. Vision (2001).
[21] F.X. Yu, R. Ji, M.-H. Tsai, G. Ye, S.-F. Chang, Weak attributes for large-scale image retrieval, in: Computer Vision and Pattern Recognition, 2012.
[22] A. Torralba, R. Fergus, Y. Weiss, Small codes and large image databases for recognition, in: Computer Vision and Pattern Recognition, 2008.
[23] J. Deng, A.C. Berg, L. Fei-Fei, Hierarchical semantic indexing for large scale image retrieval, in: Computer Vision and Pattern Recognition, 2011.
[24] S. Zhang, Q. Tian, Semantic-aware co-indexing for near-duplicate image retrieval, in: International Conference on Computer Vision, 2013.
[25] L. Xie, Q. Tian, B. Zhang, Spatial pooling of heterogeneous features for image applications, in: ACM Multimedia, 2012.
[26] H. Jegou, C. Schmid, H. Harzallah, J. Verbeek, Accurate image search using the contextual dissimilarity measure, in: IEEE Transactions on Pattern Analysis and Machine Intelligence, 2010.
[27] S. Zhang, M. Yang, T. Cour, K. Yu, D.N. Metaxas, Query specific fusion for image retrieval, in: European Conference on Computer Vision, 2012.
[28] R. Baeza-Yates, B. Ribeiro-Neto, et al., Modern information retrieval, 1999.
[29] L. Zheng, S. Wang, Z. Liu, Q. Tian, Lp-norm IDF for large scale image search, in: Computer Vision and Pattern Recognition, 2013.
[30] O. Chum, M. Perdoch, J. Matas, Geometric min-hashing: finding a (thick) needle in a haystack, in: Computer Vision and Pattern Recognition, 2009.
[31] W. Zhou, H. Li, Y. Lu, Q. Tian, Large scale image search with geometric coding, in: ACM Multimedia, 2011.
[32] W. Zhou, H. Li, Y. Lu, Q. Tian, SIFT match verification by geometric coding for large-scale partial-duplicate web image search, in: ACM Transactions on Multimedia Computing, Communications, and Applications, 2013.
[33] S. Zhang, Q. Huang, G. Hua, S. Jiang, W. Gao, Q. Tian, Building contextual visual vocabulary for large-scale image applications, in: ACM Multimedia, 2010.
[34] Y. Zhang, Z. Jia, T. Chen, Image retrieval with geometry-preserving visual phrases, in: Computer Vision and Pattern Recognition, 2011.
[35] Y.-H. Kuo, K.-T. Chen, C.-H. Chiang, W.H. Hsu, Query expansion for hash-based image object retrieval, in: ACM Multimedia, 2009.
[36] O. Chum, A. Mikulik, M. Perdoch, J. Matas, Total Recall II: query expansion revisited, in: Computer Vision and Pattern Recognition, 2011.
[37] R. Fergus, P. Perona, A. Zisserman, A visual category filter for google images, in: European Conference on Computer Vision, 2004.
[38] M. Donoser, H. Bischof, Diffusion processes for retrieval revisited, in: CVPR, 2013.
[39] G.-J. Qi, C. Aggarwal, Q. Tian, H. Ji, T.S. Huang, Exploring context and content links in social media: a latent space method, in: IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012.
[40] K. Heath, N. Gelfand, M. Ovsjanikov, M. Aanjaneya, L.J. Guibas, Image webs: computing and exploiting connectivity in image collections, in: Computer Vision and Pattern Recognition, 2010.
[41] B.J. Frey, D. Dueck, Clustering by passing messages between data points, 2007.
[42] S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine, in: Computer Networks and ISDN Systems, 1998.
[43] M. Buckland, F. Gey, The relationship between recall and precision, J. Am. Soc. Inform. Sci. (1994).