Edgel Index for Large-Scale Sketch-based Image Search*

Yang Cao[1], Changhu Wang[2], Liqing Zhang[3], Lei Zhang[2]
[1,3]MOE-Microsoft Key Laboratory for Intelligent Computing and Intelligent Systems, Shanghai Jiao Tong University
[2]Microsoft Research Asia, Beijing, China
[1]cybersun.sjtu@gmail.com, [2]{chw, leizhang}@microsoft.com, [3]zhang-lq@cs.sjtu.edu.cn

Abstract

Retrieving images to match with a hand-drawn sketch query is a highly desired feature, especially with the popularity of devices with touch screens. Although query-by-sketch has been extensively studied since 1990s, it is still very challenging to build a real-time sketch-based image search engine on a large-scale database due to the lack of effective and efficient matching/indexing solutions. The explosive growth of web images and the phenomenal success of search techniques have encouraged us to revisit this problem and target at solving the problem of web-scale sketch-based image retrieval. In this work, a novel index structure and the corresponding raw contour-based matching algorithm are proposed to calculate the similarity between a sketch query and natural images, and make sketch-based image retrieval scalable to millions of images. The proposed solution simultaneously considers storage cost, retrieval accuracy, and efficiency, based on which we have developed a real-time sketch-based image search engine by indexing more than 2 million images. Extensive experiments on various retrieval tasks (basic shape search, specific image search, and similar image search) show better accuracy and efficiency than state-of-the-art methods.

1. Introduction

Shape plays an important role in human visual perception, and has been widely used as a basic representation for a variety of computer vision tasks, such as object detection and recognition [14, 1]. As a core research problem in computer vision, searching for images to match with a hand-drawn sketch query has become a highly desired feature, especially due to the explosive growth of web images and the popularity of devices with touch screens. An effective and efficient technique for sketch-based image search technique could enable many useful applications, such as enhancing traditional keyword-based image search, and enlightening children/designers' drawing.

Sketch-based image search has been extensively studied since 1990s. However, due to the lack of an efficient index solution, it still remains very challenging to develop a

*This work was performed at Microsoft Research Asia.

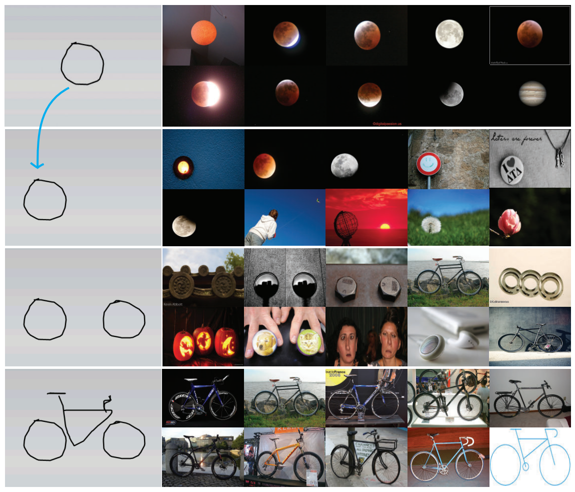
Figure 1. Example results of the MindFinder system, queried by some sequential interactive operations. For each step, top 10 results from the database of 2 million images are shown. The user first drew a circle-like shape, and then progressively added more curves. As we can see, every newly added stroke leads to more impressive and relevant results.

sketch-based search system for millions or even billions of images. Most research efforts still focus on the study of sketch-to-image matching on a small-scale dataset. But in practice, a large-scale image database is highly desired to ensure the system can always find good matches for any sketch query. The necessity of an efficient index solution further makes sketch-based image search more challenging.

To build a large-scale sketch-based image search engine, we need to overcome the following two challenges, i.e. *matching* and *indexing*. However, the two challenges are intimately coupled with each other, making it inappropriate to just study one of them. The complexity of a matching algorithm determines whether a proper index structure can be designed to speed up the retrieval process. Meanwhile, the growing desire of searching in larger databases poses a more rigid requirement on the matching precision, for a larger database increases the possibility of returning false positives in top results.

In computer vision, many research efforts have been spent on the shape-to-image matching problem. To bridge the representational gap, many methods first generate some intermediate descriptors, e.g. edge histogram [7], or directly extract representative contours, e.g. Canny Edge [3], from natural images in the database. Other methods try to avoid this problem and constrain themselves to searching a special kind of artworks, such as clip arts [15, 8] or simple patterns [11, 12], which cannot be easily generalized to natural images. To facilitate the similarity calculation between a sketch query and a natural image, mainstream approaches divide them into the same number of blocks in a 2-D space [7] or angular sectors in a polar space [5], in which the final representations are all 1-D vectors with equal length. This process greatly speeds up the pairwise matching. However, none of these methods can fully capture the spatial information of contours within each block. The other trend is to use complex models to encode geometric information and mutual relationship of objects, e.g. topology models [15, 8]. However, these models are computationally very expensive.

In contrast, little work has been done for solving the shape-based indexing problem. Most existing query-by-sketch methods suffer from the scalability issue due to the lack of efficient indexing mechanisms. When scaling up to millions of images, the response time and system cost are generally unacceptable. A few methods [7] in the literature were reported to increase the data size to more than one million. However, their core techniques merely rely on linear scan in the whole database, which greatly limits their scalability to larger image corpora. In the industry, Gazopa[1] is a large-scale multi-modal image search engine, whose technical details of the sketching part are undisclosed. However, its response time for a sketch query is typically more than 15 seconds, which is far from satisfactory.

In this work, we systematically investigate the problem of large-scale sketch-based image search, and propose an efficient matching/indexing framework targeting at million-level or even larger image corpora. First, a novel index structure called edgel[2] index is proposed for sketch-based image search by converting a shape image to a document-like representation. Different from the well-known *bag-of-features* representation in local feature-based image retrieval, where the visual vocabulary is quantized in the visual space, we describe a *visual word* using a triple (x, y, θ) of the position $\mathbf{x} = (x, y)$ of an edge pixel (*edgel*) and the edgel orientation θ at that position. By converting $< image, edgels >$ representation to $< document, words >$ representation, we can leverage an inverted index-like structure to speed up the sketch-based image search and make real-time response possible in a million-level database. Second, we propose a matching algorithm called *structure-consistent sketch matching* to measure the similarity between a sketch query and a database image, which could be efficiently implemented by our index strategy. Third, based on the proposed matching/indexing framework, we build a sketch-based image search engine called MindFinder[3] [4], which indexes 2.1 million Flickr images with 6.5GB memory cost on a common server, and supports real-time response (around 1 second). See Fig.1 for example search results.

[1]http://www.gazopa.com/

[2]edgel - the abbreviation of edge pixel

[3]http://research.microsoft.com/en-us/projects/mindfinder/

It is worth noting that the tags of web images can be used to bridge the semantic gap between a query sketch and a natural image in case the sketch is insufficient to describe a user's search intent. MindFinder also supports adding tags to associate with the sketch query to further enhance the relevance of search results.

It should be noted that there are two works that looks related but are different from this work. Retrievr[4] is an interesting image search engine that enables users to find images by drawing color strokes, which actually matches the color distributions rather than shapes. Sketch2Photo [6] has a sketch input interface, but it is an image montage system, in which the sketch works as a rough filter to extract candidate objects from a small image set collected by tag queries. It often takes long time (in minutes) to composite an image.

2. Edgel Index

In this section, we introduce the proposed indexing strategy for large-scale sketch-based image search, followed by the structure-consistent sketch matching algorithm in the next section.

2.1. Problem Formulation

The sketch-based image search in this work is defined as follows. As shown in Fig.1, a user can draw some strokes to represent the contours of an object(s) or a scene, and our system will return the *"best matched"* images to the user. The so-called "best matched" are two folds: 1) *shape sensitive*, which means that the shape of the resulting image should be as close as possible to the user's input, and 2) *position sensitive*, which means that the matched object should be at a similar position as the input sketch. In this paper, a sketch is represented by multiple contours, and each contour consists of many edgel pixels (*edgels*). For each image in the database (called database image), we first convert it to a shape image by edge detection and boundary detection, and then we compare the similarity between a query sketch and each database image in the shape space. We will detail the image feature extraction in the experiment section.

To achieve this goal, many similarity measures could be adopted, among which we choose the *Chamfer Matching* (CM) [2] and its variants, e.g. the Oriented Chamfer Matching (OCM) [16], due to their good performance on comparing contours of objects [10]. Let us use a set of edgels $\mathcal{D}$ to represent the contours of an image, in which the position of an edgel $p \in \mathcal{D}$ is denoted by $\mathbf{x}_p = (x_p, y_p)$ and its gradient orientation is denoted by θ_p. The basic Chamfer Distance [2] from a database image $\mathcal{D}$ to the query sketch $\mathcal{Q}$ is defined as follows:

$$Dist_{\mathcal{D}\to\mathcal{Q}} = \frac{1}{|\mathcal{D}|}\Sigma_{p\in\mathcal{D}} \min_{q\in\mathcal{Q}} \|\mathbf{x}_p - \mathbf{x}_q\|_2 \quad (1)$$

[4]http://labs.systemone.at/retrievr

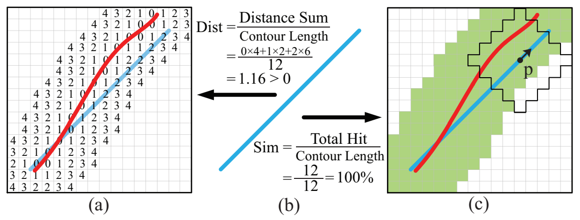
Dist = Distance Sum / Contour Length
= 0×4+1×2+2×6 / 12
= 1.16 > 0
Sim = Total Hit / Contour Length
= 12/12 = 100%
p
(a)
(b)
(c)

Figure 2. Illustration of how to compare an object contour (blue line) with a query sketch (red line). For convenience, we adopt the L_1 distance and only show one orientation channel for each method. (a) A *distance transform* map generated from the sketch using the OCM method. (b) The object contour. (c) A *Hit* map generated from the sketch using the proposed IOCM method.

where $|\mathcal{D}|$ is the number of edgels of image $\mathcal{D}$. Chamfer Matching seeks to find the nearest edgel on the query sketch $\mathcal{Q}$ for every edgel of the database image $\mathcal{D}$. In order to reduce its complexity from $O(|\mathcal{D}|\times|\mathcal{Q}|)$ to $O(|\mathcal{D}|)$, a *distance transform* map (as illustrated in Fig.2) of the query sketch $\mathcal{Q}$ could be constructed in advance, which actually uses storage cost to reduce time cost. The symmetric Chamfer Distance is given by:

$$Dist_{\mathcal{Q},\mathcal{D}} = \frac{1}{2}(Dist_{\mathcal{Q}\to\mathcal{D}} + Dist_{\mathcal{D}\to\mathcal{Q}}) \tag{2}$$

To encode orientation information of the edgel during the matching, the basic Oriented Chamfer Matching [16] was proposed:

$$Dist_{\mathcal{D}\to\mathcal{Q}} = \frac{1}{|\mathcal{D}|}\Sigma_{\theta\in\Theta}\Sigma_{p\in\mathcal{D}\&\theta_p=\theta}\min_{q\in\mathcal{Q}\&\theta_q=\theta}\|\mathbf{x}_p - \mathbf{x}_q\|_2 \tag{3}$$

where Θ is the set of quantified orientations.

A major shortcoming of the Oriented Chamfer Matching (OCM) is its scalability issue. With OCM, we have to compare a query sketch with all the images in a database and store their distance maps for speedup. Although it is not difficult to calculate the distance between two contours using OCM method according to Eqn.3, it is not trivial to design an efficient indexing mechanism for OCM in sketch-based image retrieval, which makes chamfer matching-related methods rarely used in large-scale applications. To search within two million images, all distance transform maps for distance calculation need an extra 447GB[5] memory space, which is difficult to handle using a common server with 8GB∼32GB memory. Due to the lack of indexing mechanism, even if there exists a super machine which could load such a huge amount of data into memory, the time cost of linear scanning the whole database is still unacceptable.

Therefore, to build a practical sketch-based image search engine with OCM as the similarity matching function, we need to develop an efficient index structure for it.

[5]We assume that all images are downsampled to a size whose longest side is 200 (see Section 4.2 for details). Supposing a distance transform map has 6 orientation channels, the total size is $200 \times 200 \times 6 \times 2/1024/1.024^2 \approx 447$GB.

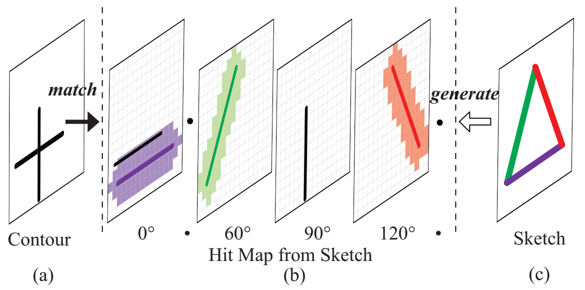

Figure 3. Toy example of the IOCM process from a contour to a sketch with tolerance radius $r = 3$. (a) The contour (black lines). (c) The sketch (colored lines). It is divided into 3 parts based on their orientations. Different colors, i.e. purple (0^o), green (60^o) and red (120^o), indicate different orientations. (b) Hit map (6 channels) generated from the sketch. In each channel, light colored grids show the valid area expanded from the corresponding sketch. We also display the contours (black lines) on corresponding channels. Two channels are ignored since they are empty. In this case, the horizontal line of the contour matches with the query sketch, whereas the vertical line does not have a match.

2.2. Edgel Index Algorithm

2.2.1 Indexable Oriented Chamfer Matching

Actually, as shown in Fig.2(a), the distance transform map of a contour is a multi-value distance map and not easy to index. To utilize an inverted index-like structure, we propose to transform the distance map to a binary similarity map (called Hit map). In our algorithm, the Hit map $M^{\mathcal{Q}}$ of a query sketch $\mathcal{Q}$ has N_Θ channels, and each channel is a binary map $M^{\mathcal{Q}}_\theta, \theta\in\Theta$, where N_Θ is the number of quantified orientations. The value of the position $\mathbf{x}_p$ on the map $M^{\mathcal{Q}}_\theta$ is given by the following Hit function:

$$Hit_{\mathcal{Q}}(p) = \begin{cases} 1 & \exists q\in\mathcal{Q}\ (\|\mathbf{x}_q - \mathbf{x}_p\|_2 \le r\ \&\ \theta_p = \theta_q), \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

where r is the *tolerance radius*.

Moreover, for an edgel $p\in\mathcal{D}$, we can also use Eqn.4 to identify whether there is an edgel in $\mathcal{Q}$ with the same orientation around position $\mathbf{x}_p$. By a simple Breadth-First-Search algorithm, the Hit map of $\mathcal{Q}$ could be generated in $\Omega(|\mathcal{Q}|)$. To calculate the similarity from image $\mathcal{D}$ to $\mathcal{Q}$, we only need a linear scan as follows:

$$Sim_{\mathcal{D}\to\mathcal{Q}} = \frac{1}{|\mathcal{D}|}\Sigma_{p\in\mathcal{D}}Hit_{\mathcal{Q}}(p) \tag{5}$$

whose time complexity is $\Omega(|\mathcal{D}|)$. Thus, the symmetric Indexable Oriented Chamfer Matching (IOCM) is given by:

$$Sim_{\mathcal{Q},\mathcal{D}} = (Sim_{\mathcal{Q}\to\mathcal{D}}\cdot Sim_{\mathcal{D}\to\mathcal{Q}})^{\frac{1}{2}} \tag{6}$$

In this way, we naturally convert the distance measurement into a similarity measurement. Compared with OCM, IOCM greatly simplifies the computational complexity and reduces the system cost. Besides the indexable property, another advantage of this measurement is the ability of tolerating local distortions. For example, as shown in Fig.2, when a user tries to draw a straight line, due to the trembling of user's fingers, local distortions (red line) are unavoidable.

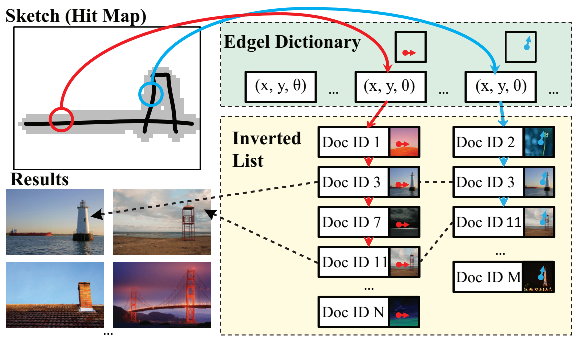

Figure 4. Index structure for database-to-query IOCM process. Picture in the upper-left corner is a mixed map combined with the user sketch and its Hit map (all channels are merged into one map for better visibility).

Using OCM (Fig.2(a)), the distance from the desired contour (blue line) to the sketch is 1.16, which is much larger than 0. While using IOCM (Fig.2(c)), the two contours are treated to be fully matched. Another example is shown in Fig.3, where the horizontal line of the contour matches with the query sketch, while the vertical line does not match.

2.2.2 Indexing for Database-to-Query Matching

Given a query sketch and its Hit map, although Eqn.5 is computationally simple, to enumerate edgels of all images from a large database is still very time consuming. Motivated by the success of search techniques, we adopt an inverted index strategy to implement the IOCM formula.

From Fig.3 we find that, in the matching process, an edgel $p = (x, y, \theta)$ from an image could have a hit only when the value of position (x, y) in channel θ of the Hit map of the query sketch is 1. Thus, if considering each edgel as a "word", we could build an edgel dictionary for the whole database, in which each entry is represented by a triple (x, y, θ). In this work, the resolution of our sketch panel is 200×200, and the orientation space is equally quantified into 6 bins, i.e. $-15^\circ \sim 15^\circ, 15^\circ \sim 45^\circ, \ldots, 135^\circ \sim 165^\circ$. Thus, our dictionary has $200 \times 200 \times 6 = 240{,}000$ entries. Based on this, we propose a novel edgel index structure (Fig.4) to organize all database images. For each entry (x, y, θ) in the dictionary, there is an inverted list of images (IDs) which contain edgel $p = (x, y, \theta)$.

With the help of this index structure, we can design an efficient ranking algorithm to quickly execute the $\mathcal{D} \rightarrow \mathcal{Q}$ matching. As shown in Fig.4, given a query sketch and its pre-generated Hit map, for each non-zero element at position (x, y) in channel θ of the Hit map, the algorithm visits its corresponding entry and inverted list in the index structure, and then goes through all the image IDs. Each ID contributes 1 hit to the similarity score of the corresponding image. Then the algorithm analyzes every non-zero element, and sums up hit numbers for each image. Finally, by dividing the number of total edgels $|\mathcal{D}|$ for each image as in Eqn.5, we can rank database images based on these normalized similarity scores.

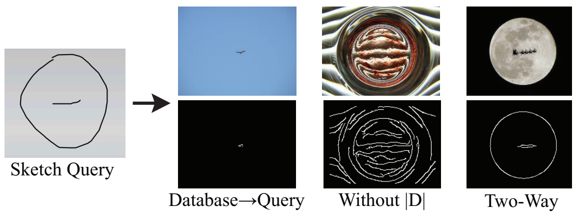

Figure 5. Top one results from three matching methods. Images in the bottom line are the corresponding contour maps.

The proposed edgel index and rank strategy fully implements the IOCM process (database to query part), which greatly reduces the time complexity from $\Omega(T)$ (T is total number of edgels in the database) to $\Omega(P \cdot L)$, where P is the number of non-zero elements in the Hit map and L is the average length of inverted lists. A typical operation cost is small (less than 1 second in this work to search more than 2 million images), for it inherently avoids unmatched edgels. The storage cost of this index structure in this work is 3.7GB[6] for 2.1 million images, which is generally acceptable for a common server.

2.2.3 Efficient Indexing for Two-Way Matching

As shown in Fig.5, the one-way $\mathcal{D} \rightarrow \mathcal{Q}$ IOCM often leads to trivial results (see the left column in Fig.5). This problem cannot be trivially solved by abolishing the denominator $|\mathcal{D}|$ in Eqn.5. Without this punishment, the top images tend to be full of edgels (the middle column). Actually, these unsatisfactory results could be filtered out by combining the opposite direction matching $\mathcal{Q} \rightarrow \mathcal{D}$ as in Eqn.6 (the right column).

The $\mathcal{Q} \rightarrow \mathcal{D}$ matching could be achieved by using a similar index mechanism as that for $\mathcal{D} \rightarrow \mathcal{Q}$ matching. To build such index structure, we should first generate Hit maps for all images in the database, which will cost about 55.9GB memory. Apparently, it cannot be handled by a common server. To address this problem, we only store raw curves instead of the distance transform map for each database image. Our solution is to first choose top N candidate images based on the database-to-query indexing[7], then online generate the distance transform maps for the N images, and finally use Eqn.6 to rank these images. In this work, N is set to 5000 for considering both time cost and retrieval precision.

Using the proposed Edgel Indexing framework, we have built the MindFinder system which indexed 2.1 million images. The total memory cost includes two parts, 3.7GB for indexing and 2.8GB for raw curve features, which are 6.5GB in total. The retrieval time is around $1s$. In Table.1, we list the resource cost after each step. The last column shows a practical solution for a large-scale sketch-based system.

[6]The database contains 1001 million individual edgels. To index them, each ID takes 4 byes, and total size is $1001 \times 4/1024/1.024^2 \approx 3.7$GB.

[7]In order to increase the recall of the candidate images, in implementation, we use $\sqrt{|\mathcal{D}|}$ instead of $|\mathcal{D}|$ in Eqn.5 in this step.

	OCM	SSM	EI	Final
Memory ($\mathcal{D} \to \mathcal{Q}$)	2.8	2.8	3.7	3.7
Memory ($\mathcal{Q} \to \mathcal{D}$)	447	55.9	55.9	2.8
Time ($\mathcal{D} \to \mathcal{Q}$)	N/A	N/A	0.8	0.8
Time ($\mathcal{Q} \to \mathcal{D}$)	N/A	N/A	N/A	0.4

Table 1. Overview of the resource cost after each step for indexing and searching 2 million images. The first two rows are memory cost (GB) and the rest are response time (second). N/A indicates it is far from practical. "EI" means SSM with edgel index (EI) strategy. "Final" means approximate two-way matching.

3. Structure-Consistent Sketch Matching

When searching images, users may draw multiple strokes to find images with multiple objects. Sometimes, the database may be lack of such ideal images. As shown in Fig.6, image A with one part well matched with the query is ranked higher than image B with two parts matched but both of them are not matched very well. However, users may prefer image B since their search intent (sketch) consists of two objects.

To handle this problem, we need to calculate the matching similarity in a global way. In the implementation, we decompose one query sketch into multiple sub-queries, and use the geometric mean of similarity scores from all these sub-queries as the final score. The order and spatial information of strokes recorded by our interface could naturally guide us to divide the sketch query into isolated sub-queries. Besides, we further divide each stroke into several continuous contours in case users prefer non-stop drawing. We consider each continuous contour with length more than L_d (half of the radius of the canvas in this work) as a sub-query, and contours less than this threshold will be counted with the next contour until it meets the requirement. For a sketch query $\mathcal{Q}$ and its N components $\{\mathcal{Q}_1, \mathcal{Q}_2, ..., \mathcal{Q}_N\}$, the structure-consistent similarity from $\mathcal{Q}$ to $\mathcal{D}$ is given by:

$$Sim^{SSM}_{\mathcal{Q}\to\mathcal{D}} = (\Pi_{i=1}^{N} Sim_{\mathcal{Q}_i\to\mathcal{D}})^{\frac{1}{N}} \tag{7}$$

which guarantees that each part is evenly considered and the global structure becomes more important than local details. In implementation, if one component is mismatched, we assume there is still one hit to avoid zero similarity. It should be noted that, the time complexity of structure-consistent sketch matching (Eqn.7) is still $\Omega(|\mathcal{Q}|)$, for it just linearly scans the Hit map as $Sim_{\mathcal{Q}\to\mathcal{D}}$ does, and separately counts similarity for each sub-queries as they are individual ones. Thus, the final Structure-consistent Sketch Matching (SSM) score is given as follows:

$$Sim^{SSM}_{\mathcal{Q},\mathcal{D}} = (\Pi_{i=1}^{N_\mathcal{Q}} Sim_{\mathcal{Q}_i\to\mathcal{D}})^{\frac{1}{2N_\mathcal{Q}}} \cdot (\Pi_{j=1}^{N_\mathcal{D}} Sim_{\mathcal{D}_j\to\mathcal{Q}})^{\frac{1}{2N_\mathcal{D}}} \tag{8}$$

where $Sim_{\mathcal{Q}_i\to\mathcal{D}}$ and $Sim_{\mathcal{D}_j\to\mathcal{Q}}$ are given by Eqn.5.

For easy implementation, we only use SSM in $\mathcal{Q} \to \mathcal{D}$ matching, which does not influence the proposed Edgel Index structure.

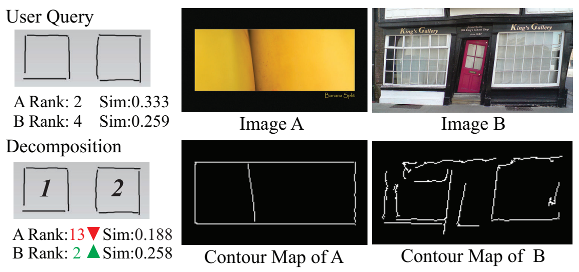

Figure 6. Illumination of structure-consistent matching. It shows ranks and similarity scores before and after adding structure-consistent matching. As we can see, image B is ranked higher than A after taking account of the query structure.

4. Experimental Results

In this section, we evaluate the proposed algorithms and framework for three tasks: basic shape search, specific image search, and similar image search.

4.1. Experiment Setup

To evaluate the proposed matching and indexing algorithms, we built a sketch-based image search system which indexed 2.1 million images crawled from Flickr.

To the best of our knowledge, the *Tensor Descriptor* (denoted by TENSOR in this work) proposed by Eitz *et al.* [7] is the only published large-scale sketch-based image search work in the literature. Different from our system, TENSOR is a descriptor-based method rather than a raw contour-based method. Moreover, no index system is particularly designed in TENSOR and it has to linearly scan the database to respond a query.

We also evaluated three variants of the proposed approaches: the proposed edgel index framework with *structure-consistent* matching (denoted by EI-S), the database-to-query *one-way* matching/indexing without structure-consistent matching (EI-1) and the *two-way* matching/indexing without structure-consistent matching (EI-2).

4.2. Image Preprocessing

Before extracting contour features, we first downsample images to a small size (maximal side 200) to keep good balance between structure information preservation and storage cost. Then, we adopt the Berkeley detector [13] to extract object contours. With this detector, a natural image is transformed into a contour map, and each contour is composed by edgels.

4.3. Basic Shape Search

In this part, we evaluate our framework by using eight elemental shapes as sketch queries (see Fig.7 for shape information). These queries are fundamental components to compose most of complex sketches, and this experiment can verify whether our approach can precisely match structures and successfully present users' intents.

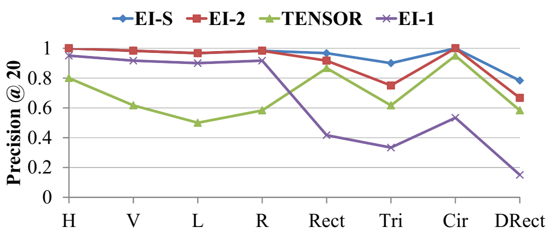

Figure 7. Performance comparison for basic shape search. In abscissa there are Horizontal line (H), Vertical line (V), Left diagonal (L), Right Diagonal (R), Rectangle (Rect), Triangle (Tri), Circle (Cir) and Double Rectangle (DRect).

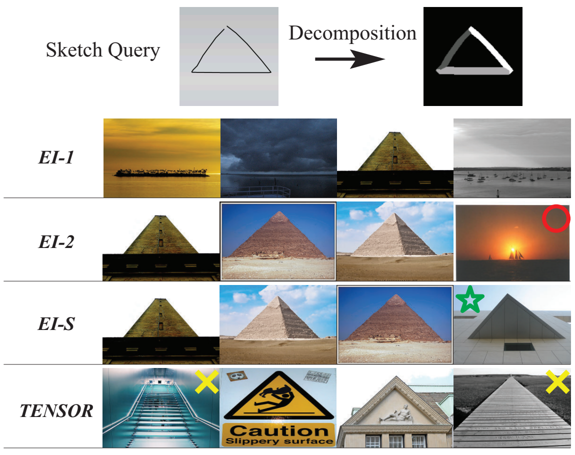

Figure 8. Illustration of top 4 images retrieved from different methods. The user sketch and its decomposition (different gray-scale indicates different sub-queries) are provided in the top line.

For each shape, we have four queries, out of which one was automatically generated by computer, and the others were freely drawn by three subjects. Then the three subjects were asked to evaluate the results of each method by labeling top 20 results as relevant or not. The evaluation criterion is whether the salient object contours of the image is highly matched with the query sketch.

The precision of the top 20 search results for each shape is shown in Fig.7. It is clear that EI-S is superior than other three methods due to its good accuracy on matching shapes. The performance of EI-2 is quite close to that of EI-S except for the multi-line shapes. The major reason is that, EI-2 only counts the total matching edgels without considering the structure information, whereas EI-S could preserve the global structure to some extend. As shown in Fig.8, EI-S divides the query sketch into 3 sub-queries (presented by the grey-scale image), and the results are all triangles with similar structure, including an unsymmetrical triangle (marked by a green star). However, EI-2 misses this image at the top 4 place, instead, returning an image with only a long bottom line (marked by a red circle) partially matched with the initial query.

From Fig.7 we can also see that TENSOR performs worse than both EI-S and EI-2 when searching elemental shapes, especially for basic lines. This is caused by the inherent limitation of descriptor-based method, which assigns a short vector to a region and loses part of local spatial information. In contrast, our raw contour-based methods could fully depict the original shape information. As shown in Fig.9, for TENSOR (right), though the corresponding areas are roughly related with the user sketch, few of our subjects treated it as relevant, while the result from EI-S (left) is highly matched. The weakness at distinguishing a specific line from local texture with similar orientation distribution, could explain why two images of pathway (marked by yellow crosses) are retrieved in topmost results of TENSOR in Fig.8. On the contrary, EI-1 prefers to retrieve images with simple shapes, and thus is difficult to handle complex queries. The inferior results of EI-1 in the other two tasks also verified this point (see Fig.10 and 11 in the next two subsections).

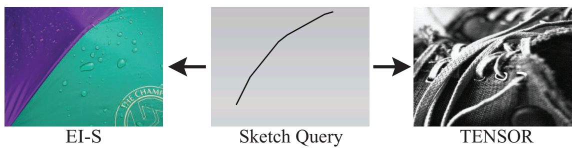

Figure 9. Top one results of the EI-S method (left) and tensor method (right) when searching an oblique line (middle).

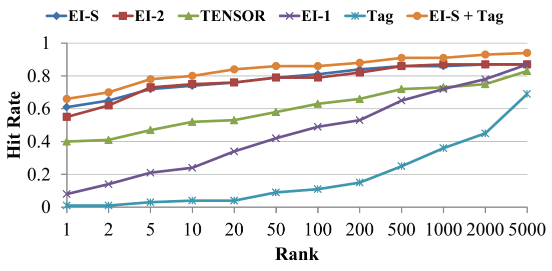

Figure 10. Performance comparison for specific image search.

We conducted our experiments on an Intel Xeon 2.4GHz Quad-Core server with 16GB memory. The memory cost of EI-S are 6.5GB, and the average response time is 0.8 second; as to TENSOR method, the costs are 12.6GB and 2.1 seconds respectively. These results have shown the importance of an index scheme in a large-scale database.

4.4. Specific Image Search

In this task, we want to evaluate the performance of our system by finding a target image using a sketch query. We invited five subjects to finish 100 search tasks. For each task, we use one image as the query reference. Thus, there are totally 100 target images, in which 20 images of common objects were manually selected, and the others were randomly picked from the database. For each reference image, each subject was asked to first carefully watch it, and then search it by sketching its major contours. Besides, subjects were also asked to add tags for every sketch query to facilitate their search. This approach is denoted as EI-S(+Tag). The pure tag-based method (denoted by Tag) by using the above additional tags as the query is also evaluated.

"Hit Rate@K", which is the proportion of all the 100 search tasks that could rank the target image in the top K

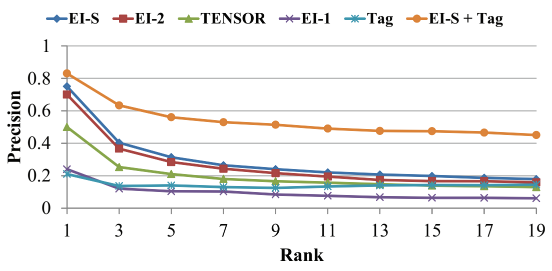

Figure 11. Performance comparison for similar image search.

search results, is defined as the measurement.

The results are shown in Fig.10, from which we can see that, EI-S is useful for finding specific images, since 61 target images are ranked at top one, which is 10.9% higher than that of EI-2.

The performance of Tag is not good, and only 4 targets are ranked in the top 20 results. The reason of the poor performance of tag queries is two folds. On the one hand, the target image might not contain the query tag(s). In this case, the target image cannot be found at all if only using a tag query. On the other hand, even if the subject chooses a right tag, the tag feature is not distinguishable enough to locate a specific image in the top results. Thus, from this experiment we can see that, pure tag feature is not applicable to specific image search task. However, as a complement of sketch-based search, tag plays a very important role in reducing the semantic gap. As shown in Fig.10, EI-S(+Tag) performs better than EI-S (8.2% better in terms of Hit Rate@1).

As we can see in Fig.10, TENSOR is worse than EI-S and EI-2 methods. As to the Hit Rate@1, EI-S and EI-2 are 52.5% and 37.5% higher than TENSOR. This result is consistent with our previous observation that TENSOR is incapable of both precisely depicting local structure and matching two contours. It also verified our assumption that in such a huge database, raw contour-based method for accurate matching is very necessary.

The average response time of EI-S for 100 complex sketch queries is about 1.17s, which is about 3 times faster than TENSOR's. Moveover, the average response time of EI-2 (1.16s) is almost the same as EI-S, which shows that the consistent-structure matching does not bring much time cost comparing with EI-2.

4.5. Similar Image Search

In real scenarios, users may be interested in searching a concept in mind by sketching and tagging. This task is termed as "similar image search" in this work. Notice that the definition to "correctness" for the new task is not as rigorous as that for the specific search task.

For convenience, we reused the experimental setup of specific image search. The subjects were asked to evaluate the precision of top N results for each query rather than only evaluate the hit of the target image. For all methods in our comparison, the criterion of relevance is not only structurally (in terms of shape) but also semantically (in terms of concept) matched with the target image. The comparison results are given in Fig.11.

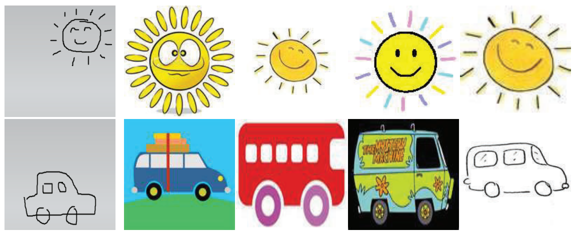
Figure 13. Top four results of sketch-based clipart image search. The system is scale and translation invariant to query sketch. Images in the first column are initial sketch queries.

From Fig.11 we can see that, by relaxing the constraint from finding specific images to finding similar objects with the same structure, for 75% of the 100 queries, EI-S successfully ranks the correct images at the top one. By combining with tags to reduce the semantic gap, the performance of returning similar images at the first position increases to nearly 83%. For the tag-only search, the poor performance can be explained as the lack of shape and structure information in the search process. Although the TENSOR method also achieves performance improvement in this task, it still performs much worse than the proposed approaches.

Fig.12 shows several sketch queries and the corresponding top search results of EI-S in the 2.1 million database, in which the top three ones produced very impressive results, while the bottom three ones are selected from the query pool with "bad" search results in terms of similar image search.

5. Conclusions and Discussions

How to build a practical sketch-based image search engine is deemed to be a very challenging problem in both academic and industrial communities. In this work, we have systematically investigated this problem, and successfully built a real-time large-scale sketch-based image search engine. We have proposed the structure-consistent sketch matching algorithm as well as the edgel index structure for implementing the matching algorithm. This might be the first indexing strategy particularly designed for large-scale sketch-based image search.

In this work, we describe a visual word using a triple (x, y, θ) of the position $\mathbf{x} = (x, y)$ of an edge pixel (*edgel*) and the edgel orientation θ at that position. The introducing of edgel position to visual word description enables a highly efficient inverted index structure and makes it possible to build a real-time large-scale sketch-based image search system. Although the proposed solution can tolerate local distortions of users' sketch inputs (See Figure 2(c) for details), this representation inevitably results in the loss of position-invariant feature for sketch-to-image matching, which is often a desired feature for query sketching. Technically it is very difficult to get a tradeoff solution between efficient index and position-invariant matching. We have noticed that most position-invariant works [9, 14] usually

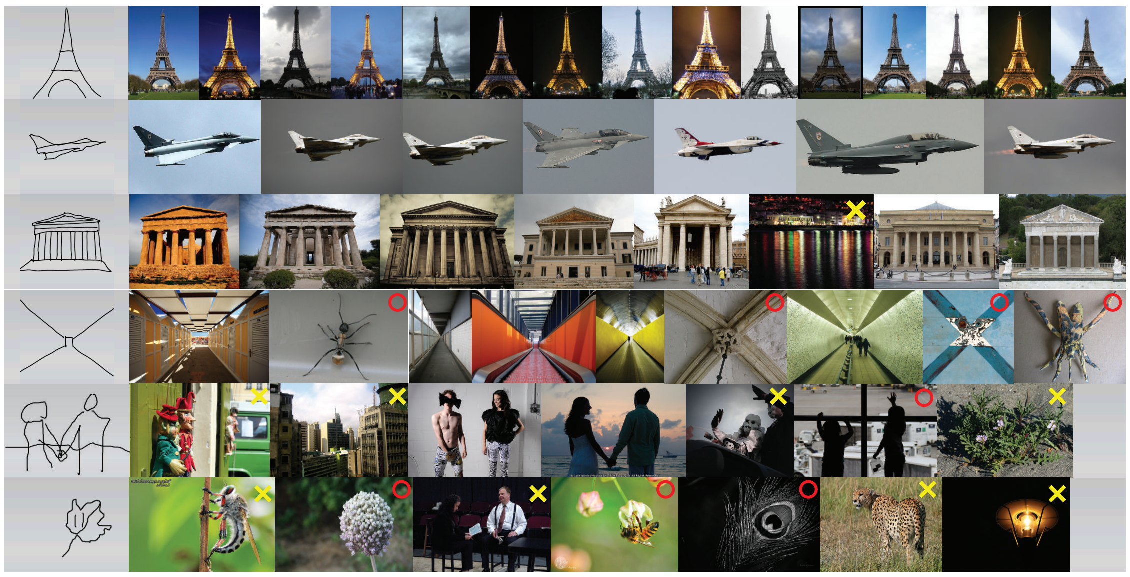
Figure 12. Example queries and the corresponding top results. According to subjects' labeling, images marked by yellow cross indicate that they are irrelevant with the corresponding sketch query in terms of both structure and conceptual meaning; while images marked by red circle mean that although they are not the same concept of the sketch query, these images still have very similar structures to the query.

target at searching images in a small-scale dataset. Such methods usually require a complex algorithm to find objects in arbitrary positions in an image. The complexity of these algorithms makes it impractical to design a feasible index structure to speed up the search process, which constrains them only to small-scale retrieval tasks.

In practice, a large-scale database is highly desired to serve for users' various search intents. We expect that an efficient and scalable index solution can compensate the lack of position-invariant matching, as in a large-scale search system, users care more about search precision than recall. If the database is large enough, we can always find good matches from the database and return them to users. Works in this genre consider the layout of the drawings in the query panel as a constraint, and thus could design simpler matching algorithms to search images in a web-scale database.

We can also make further assumptions in some vertical domains, such as clipart image search in which we can assume that there is only one main object in a clipart image. Based on this assumption, a variant of the proposed system was developed to support scale and translation-invariant sketch-based search in a 0.7 million clipart image set, which is much more robust to users' inputs (see Fig.13).

However, an efficient index solution capable of affine-invariant shape-to-image matching is definitely worth to be explored and we treat this as our future work.

Acknowledgements

The work of Yang Cao and Liqing Zhang was partially supported by the National Natural Science Foundation of China (Grant No. 90920014).

References

[1] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, 2002.

[2] G. Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *PAMI*, 1988.

[3] J. Canny. A computational approach to edge detection. *Readings in computer vision: issues, problems, principles, and paradigms*, 1987.

[4] Y. Cao, H. Wang, C. Wang, Z. Li, L. Zhang, and L. Zhang. MindFinder: interactive sketch-based image search on millions of images. In *ACMMM*, 2010.

[5] A. Chalechale, G. Naghdy, and A. Mertins. Sketch-based image matching using angular partitioning. *TSMC*, 2005.

[6] T. Chen, M. Cheng, P. Tan, A. Shamir, and S. Hu. Sketch2Photo: internet image montage. *TOG*, 2009.

[7] M. Eitz, K. Hildebrand, T. Boubekeur, and M. Alexa. An evaluation of descriptors for large-scale image retrieval from sketched feature lines. *Computers & Graphics*, 2010.

[8] M. J. Fonseca, A. Ferreira, and J. a. Jorge. Sketch-based retrieval of complex drawings using hierarchical topology and geometry. *Computer-Aided Design*, 2009.

[9] H. Ip, A. Cheng, W. Wong, and J. Feng. Affine-invariant sketch-based retrieval of images. In *Computer Graphics International*, 2001.

[10] K. Lee, Y. J. And Grauman. Shape discovery from unlabeled image collections. In *CVPR*, 2009.

[11] W. Leung and T. Chen. Retrieval of sketches based on spatial relation between strokes. In *ICIP*, 2002.

[12] S. Liang and Z. Sun. Sketch retrieval and relevance feedback with biased SVM classification. *PR Letters*, 2008.

[13] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 2004.

[14] J. Shotton, A. Blake, and R. Cipolla. Multiscale categorical object recognition using contour fragments. *PAMI*, 2008.

[15] P. Sousa and M. Fonseca. Sketch-based retrieval of drawings using topological proximity. In *VLC*, 2008.

[16] B. Stenger, A. Thayananthan, P. Torr, and R. Cipolla. Model-based hand tracking using a hierarchical bayesian filter. *PAMI*, 2006.