Sketch-based Image Retrieval via Shape Words

Changcheng Xiao[1]*, Changhu Wang[2]†, Liqing Zhang[1], Lei Zhang[3]
[1]Shanghai Jiao Tong University, Shanghai, China
[2]Microsoft Research, Beijing, China
[3]Microsoft Corporation, Redmond, USA
xchangcheng@gmail.com, chw@microsoft.com,
zhang-lq@cs.sjtu.edu.cn, leizhang@microsoft.com

ABSTRACT

The explosive growth of touch screens has provided a good platform for sketch-based image retrieval. However, most previous works focused on low level descriptors of shapes and sketches. In this paper, we try to step forward and propose to leverage *shape words* descriptor for sketch-based image retrieval. First, the *shape words* are defined and an efficient algorithm is designed for *shape words* extraction. Then we generalize the classic Chamfer Matching algorithm to address the *shape words* matching problem. Finally, a novel inverted index structure is proposed to make *shape words* representation scalable to large scale image databases. Experimental results show that our method achieves competitive accuracy but requires much less memory, e.g., less than 3% of memory storage of MindFinder. Due to its competitive accuracy and low memory cost, our method can scale up to much larger database.

Categories and Subject Descriptors

H.3.3 [**Information Retrieval**]: Search Process, Query formulation

Keywords

Sketch-based Image Retrieval, Shape Words

1. INTRODUCTION

Owing to the popularity of digital cameras, millions of new digital images are freely accessible online every day, which brings a great opportunity for image retrieval. Usually, users search images with text queries. But the shape and location of the object are hard to be formulated with a few keywords. Thus, query-by-example (QBE) was proposed. However, in QBE, the query has to be an example image, which is usually the reason of searching. As the popularity of touch-screen devices, searching images by drawing has become a highly desired feature for users, which is complementary and thus can be combined with query-by-keyword and query-by-example modalities.

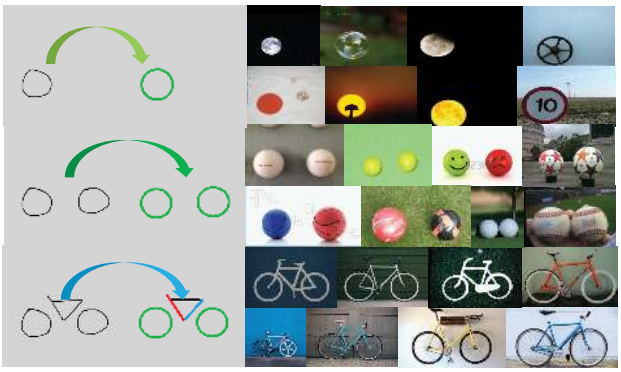

Figure 1: Example results of our system. Suppose we want to search for a bike. We may draw two circles as its wheels and draw some line segments as its frame. When we draw these sketches the system will extract shape words (line segments and circular arcs) and search similar images with these shape words.

Sketch based image retrieval (SBIR) has been extensively studied since 1990s, and stepped into large-scale scenarios in recent years. In 2010, Eitz et al. [5] built an SBIR system based on Tensor descriptors by linearly scanning the whole database for each query, which greatly limits its scalability. In 2011, Cao et al. [3] built the MindFinder system based on indexable Oriented Chamfer Matching (OCM) to solve the indexing problem of SBIR. In 2012, Zhou et al. [8] proposed a convolution based descriptor, and Kai-Yu Tseng [7] proposed to use 'HashBits' to compress the Distance Transform descriptor. In 2013, Sun et al. [6] built a billion scale SBIR system with vector-like Chamfer feature pairs.

However, most of these methods [3, 5, 6, 7, 8] focus on low level descriptors of sketches like local patches or edge pixels, which require huge amount of memory storage. In this work, we try to go one step forward and see the shapes/sketches in a higher view. Different from MindFinder [4] which indexes and matches with (sampled) edge pixels, we propose to use *shape words* to represent both the query sketch and database images. As shown in Fig. 1, when asked to draw a bike, most users will probably draw two circles as its wheels

*Changcheng Xiao performed this work while being an intern at Microsoft Research Asia.
†Corresponding author.

ICMR'15, June 23–26, 2015, Shanghai, China.
ACM 978-1-4503-3274-3/15/06.
http://dx.doi.org/10.1145/2671188.2749360.

and draw a triangle as its frame. Ideally, it will be quite powerful if we can successfully extract the triangles and circles from images, and match them with the shapes of the query sketch. However, it's still a very challenging task to extract such shapes from natural images with high precision/recall. To balance the robustness and computational complexity, we leverage a middle level descriptor between edge pixels and shapes, i.e., line segments and circular arcs, as *shape words*. We propose an efficient algorithm for *shape words* extraction, followed by a generalization of the classic Chamfer Matching algorithm [1] for *shape words*' matching. Finally, a novel inverted index structure is proposed to make *shape words* representation scalable to large scale image databases. The experimental results demonstrate the high memory efficiency and retrieval accuracy of the proposed method.

2. SHAPE WORDS REPRESENTATION

In this section, we introduce the *shape words* representation followed by its extraction algorithm. First, we convert the natural image to a stroke-like representation by salient boundary extraction and stroke decomposition. Then we extract *shape words* from these strokes. The framework of *shape words* extraction is illustrated in Fig. 2.

2.1 Shape Words Design Strategy

Our goal is to define a robust while compact descriptor to encode the precise structure information of shapes and sketches. Edge pixels are the smallest components of sketches and they are robust to local distortions[3]. However, single edge pixel only carries limited information (e.g., gradient) of sketches. What's worse, there are thousands of edge pixels in a single image, which require a large amount of memory to index. Shapes like circles and triangles are informative and compact but they are not robust in extraction and matching. Thus, we compromise between the edge pixels and real shapes, and propose *shape words*.

Definition 1. A *shape word* is a small segment constructed by a group of connected edge pixels. Different *shape words* have different properties like location, direction and size.

In this paper, we choose line segments and circular arcs as *shape words*. The *shape words* based representation is robust to represent any sketch and very compact at the same time. A line segment can be represented by a tuple $L = (x, y, l, \theta)$, in which (x, y), l, θ represent the center location, length, and direction of the line segment. A circular arc is represented by a tuple $C = (x, y, r, \theta_1, \theta_2)$, in which (x, y), r, θ_1, θ_2 are the center location, radius, and angle range of the circular arc. Based on these representations, a natural image or a user sketch can be represented by a bag of *shape words* as:

$$\overline{\mathcal{F}} = \{L_1, L_2, \dots C_1, C_2, \dots\}. \quad (1)$$

In our Flickr1M dataset, the number of *shape words* is just about 3% of the number of edge pixels.

2.2 Stroke-like Representation

Given a natural image, we first resize it to make sure that its longer side contains 200 pixels. Then we apply Canny Edge Detector [2] to extract the salient edges of the image, in which we choose proper thresholds to remove the clutter details while keep the salient edges at the same time.

In order to extract the *shape words* from user sketches and natural edge images, we have to convert them to stroke-like representations. Although we can decompose user sketches to strokes easily, it isn't a trivial task to extract strokes from edge images. However, as different stroke decomposition results will not affect the *shape words* representation too much, we extract strokes by simply considering the connectivity of edge pixels. At the end of this step, we can represent the user sketch or a natural image by a stroke set $\overline{\mathcal{S}}$.

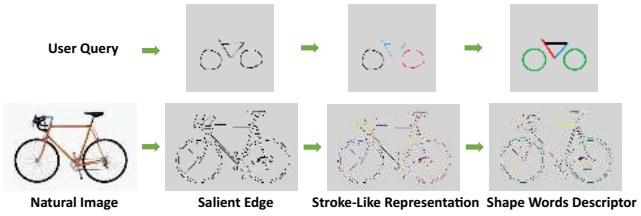

Figure 2: Framework of shape words extraction. Different colors represent different strokes or shape words.

2.3 Shape Words Extraction

First, we extract the interest point set *IP(S)* of a stroke S by calculating its polygonal approximation end points. Then we extract *shape words* set *SW(S)* by searching maximum successive interest points with least square fitting error *(LSFE)* smaller than a predefined threshold. The detail of *shape words* extraction algorithm is as follows:

Algorithm 1 Shape Words Extraction

Define:
$\overline{FL}$ is the line defined by the point F and L
$S_{A\to B}$ is a sub stroke of S from point/index A to B
$LSF, LSFE$ are the least square fitting and fitting error
$T_{split}, T_{line}, T_{arc}$ are predefined thresholds
for *Stroke* $S \epsilon \overline{\mathcal{S}}$ **do**
  Stage 1: Extract interest points set $IP(S)$
  Add its first and last point F, L to $IP(S)$
  $[P_{max}, Dist_{max}] = Argmax_P(EuclidDist_{P\to\overline{FL}}), P\epsilon S$
  if $Dist_{max} \geq T_{split}$ **then**
    Calculate $IP(S_{F\to P_{max}}), IP(S_{P_{max}\to L})$ recursively
    $IP(S) = IP(S_{F\to P_{max}}) \cup IP(S_{P_{max}\to L})$
  end if
  Stage 2: Extract shape words set $SW(S)$
  for $j = 1 \to size(IP(S)) - 1$ **do**
    $k \epsilon [j+1, size(IP(S))]$
    $Line_k = Argmax_k(LSFE_{line}(S_{i\to k}) \leq T_{line})$
    $Arc_k = Argmax_k(LSFE_{arc}(S_{i\to k}) \leq T_{arc})$
    $[L, Err_l] = LSF_{line}(S_{i\to Line_k})$
    $[C, Err_a] = LSF_{arc}(S_{i\to Arc_k})$
    if $Line_k > Arc_k | (Line_k == Arc_k \& Err_l \leq Err_a)$ **then**
      Add line segment L to $SW(S)$
    else
      Add circular arc C to $SW(S)$
    end if
    $j = Max(Line_k, Arc_k)$
  end for
end for

After all the *shape words* are extracted, the sketch query or the natural image can be represented by a bag of *shape words* as in Equation 1.

3. MATCHING AND INDEXING

In this section, we generalize the classic Chamfer Matching [1] to address the *shape words* matching problem. Then, we propose an efficient index structure for fast matching.

3.1 Shape Words Matching

Given a query sketch $\mathcal{Q}$, we want to find the best matched image(s) $\mathcal{D}$ in the database, both of which are represented by a bag of *shape words*. Thus we need to measure the similarity of $\mathcal{Q}$ and $\mathcal{D}$, denoted by $Sim_{Q,D}$.

First, we introduce the classic Chamfer Matching [1] for raw curve matching. Then we generalize the classic Chamfer Matching for *shape words* matching.

3.1.1 Classic Chamfer Matching

The classic Chamfer Matching similarity from a database image $\mathcal{D}$ to the query sketch $\mathcal{Q}$ is defined as:

$$Sim_{D\to Q} = \frac{1}{|D|}\sum_{p\epsilon D}\max_{q\epsilon Q} sim(p,q). \quad (2)$$

Chamfer Matching algorithm seeks to find the nearest pixel on the query sketch $\mathcal{Q}$ for each pixel of the database image $\mathcal{D}$. So we can define $sim(p,q)$ above as $\frac{1}{Dist(p,q)}$. Here, $|D|$ is the total number of edge pixels of $\mathcal{D}$. Similarly, we can define the similarity from $\mathcal{Q}$ to $\mathcal{D}$ as:

$$Sim_{Q\to D} = \frac{1}{|Q|}\sum_{q\epsilon Q}\max_{p\epsilon D} sim(q,p). \quad (3)$$

The symmetric Chamfer Matching similarity is given by:

$$Sim_{D,Q} = (Sim_{D\to Q}\cdot Sim_{Q\to D})^{\frac{1}{2}}. \quad (4)$$

3.1.2 Chamfer Matching for Shape Words

Chamfer Matching algorithm was actually designed for pixel-based raw curve matching. So in order to apply it to *shape words* matching, we have to generalize its matching units from pixels to *shape words* and define the similarity of *shape words*.

In the raw curve case, the Euclidean distance is applied to define the similarity of pixels. For two *shape words* p and q, we consider their type similarity $sim^t(p,q)$, location similarity $sim^l(p,q)$ and size similarity $sim^s(p,q)$. So the similarity of p and q can be defined as:

$$sim(p,q) = sim^t(p,q)\cdot sim^l(p,q)\cdot sim^s(p,q), \quad (5)$$

where each term is defined as:

$$sim^t(p,q) = \begin{cases} 1 & \text{the same type of } \textit{shape words}, \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

$$sim^l(p,q) = \begin{cases} 1 & Dist(p,q)\le R, \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

$$sim^s(p,q) = Overlap(p,q). \quad (8)$$

In Equation 7, $Dist(p,q)$ is the center distance of two *shape words* and R is the tolerance radius for $Dist(p,q)$ ($R = 15$ in our experiments). We define $Overlap(p,q)$ by considering the index structure, in which we quantize the length of *shape words* to a limited number of bins. For the same type of *shape words*, their $Overlap$ is just the maximal length they share if they are close enough.

After defining the similarity of *shape words*, we can use Equation 4 to compute the similarity score of $\mathcal{Q}$ and $\mathcal{D}$. $|D|$ and $|Q|$ are the total length of *shape words*.

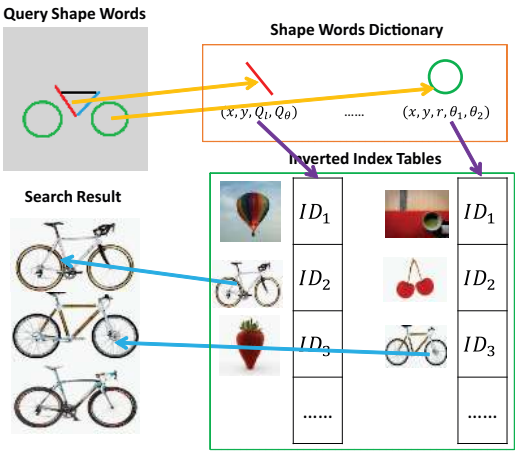

Figure 3: Shape words index structure.

3.2 Shape Words Index Structure

One of the most important design strategies for *shape words* representation and their matching algorithm is to make them indexable and thus scalable to large scale databases.

As there are two kinds of *shape words*, we build one inverted table for each of them. In order to build the inverted index tables, we need to quantize the two kinds of *shape words* first. For a line segment $L = (x,y,l,\theta)$, we quantize its directions θ to Q_θ (6 bins), i.e., $-15^\circ \sim 15^\circ$,$15^\circ \sim 45^\circ$,...,$135^\circ \sim 165^\circ$, and its length l to Q_l (20 bins). For a circular arc $C = (x,y,r,\theta_1,\theta_2)$, we quantize its radius r to Q_r (10 bins). We treat each line segment and circular arc as a *shape word* in the dictionary. As all the database images have been resized to 200×200, the dictionary size for the line segments is $200\times 200\times 20\times 6 = 4.8M$, and the dictionary size for the circular arcs is $200\times 200\times 10 = 0.4M$.

If an image ID contains a line segment $L = (x,y,l,\theta)$, a item ID will be inserted to the list at (x,y,Q_l,Q_θ). As there are much less circular arcs, we choose to store their angle range information in the inverted table. Thus if an image ID contains a circular arc $C = (x,y,r,\theta_1,\theta_2)$, an item (ID,θ_1,θ_2) will be inserted to the list at (x,y,Q_r).

For a sketch query $\mathcal{Q}$ in Fig. 3, we first extract its *shape words* and quantize them as above. Next, we propagate each *shape word* to its neighbors with distance smaller than R. Then we retrieve the inverted list of the *shape word* in these neighbors and compute their similarity score using Equation 5. Finally, we compute the two-way similarity score $Sim_{D,Q}$ with Equation 2 $\sim$ 4 and rank all the database images with their similarity scores.

4. EXPERIMENTS

To evaluate the performance of the proposed method on a large scale dataset, we built a new image dataset called Flickr1M. (1) We defined 500 object categories and used their keywords to search on Flickr and Google. Each category contains about 2K images. (2) We chose 100 meaningful categories for SBIR. Then we invited five subjects to skim through the object images and draw a sketch for each of the 100 object categories. Hence, we have $100\times 5 = 500$ sketches in total. (3) The images of other 400 categories are

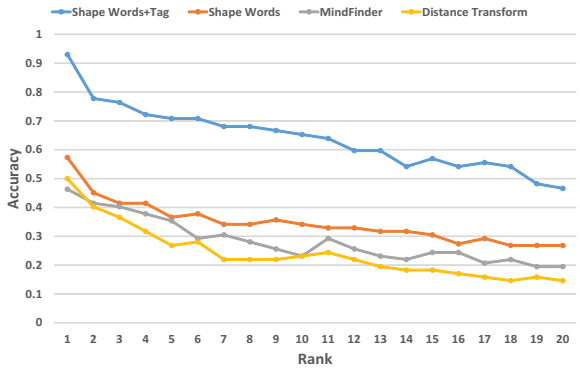

Figure 4: Retrieval accuracy on sketch-based image retrieval.

treated as the distracting images. Overall we collected 1.3M images and 500 sketches.

Then we built an SBIR system with the proposed method on this new dataset and compared its memory cost and retrieval accuracy with the MindFinder approach [3], which is one of the state-of-the-art SBIR approaches. Different from our method, MindFinder uses edge pixels for indexing and matching. We also compared with the HashBits approach [7]. As hashing and binarization will sacrifice some accuracy for efficiency, which makes Distance Transform hold the upper bound accuracy of this method. Thus, we chose to compare with Distance Transform directly.

All the experiments were conducted on a server with 2 Intel Xeon 2.66GHz Six-Core processors and 64GB memory. The average retrieval time of our system is about 1 second.

4.1 Memory Cost Saving

In Flickr1M, an image has 1361 edge pixels in average. For MindFinder approach, it needs to index about $1361 \times 1.3M = 1769.3M$ items, which costs $1769.3M \times 4Bytes \approx 6.91GB$ memory. Proper pixel sampling can reduce memory cost but may lose some accuracy. For comparison, an image has only 46.07 *shape words* in average. Thus for *shape words* method, we only need to index about $46.07 \times 1.3M \approx 59.89M$ items. As we have to encode a little more information, our system needs $240M$ memory[1] for the index structure, which is only 3.4% of MindFinder approach without sampling. MindFinder needs another 1.5G memory to store the raw curve features for two-way Chamfer Matching. So the memory cost of our system is actually less than 3% of MindFinder, which is as good as HashBits method [7].

4.2 Retrieval Accuracy

In this task, we evaluate the effectiveness of our system. We invited those 5 subjects to search their 100 sketches in the three systems: *Shape Words*, *MindFinder* and *Distance Transform*. Besides, subjects were asked to add proper tags for every sketch query to facilitate their searches. This approach is denoted by *Shape Words + Tag*. The measurement is defined as the proportion of all the 500 search tasks that could rank a similar image at rank position k.

The results are shown in Fig. 4. We can see that, *Shape Words* performs a little better than *MindFinder* and *Distance Transform*. By checking the results of each query, we found that our system performed better for those queries with obvious structures. As a negative example, a query of a bird doesn't have obvious shape structure. There will be more distortions when users draw such objects. Edge pixel-based method like *MindFinder* has better tolerance for this. Fig. 5 shows several sketch queries and the corresponding top search results.

[1] $(46.07 + 0.02 \times 3) \times 1.3M \times 4Bytes \approx 240MB$

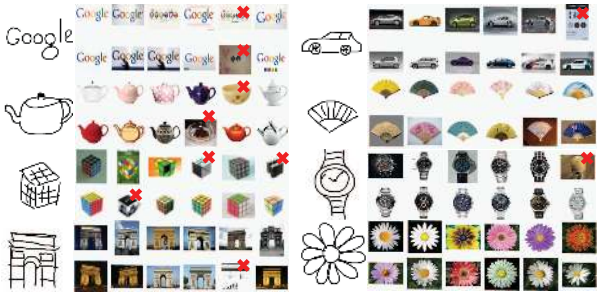

Figure 5: Example queries and corresponding results. Images marked by red cross indicate that they are irrelevant with the query.

5. CONCLUSIONS

In this paper, we proposed to use *shape words* to balance the robustness and compactness of large scale SBIR. We also designed efficient extraction, matching and indexing algorithms for the *shape words* representation. Experimental results show that our method achieves competitive accuracy but requires much less memory.

6. ACKNOWLEDGMENTS

The work of first and third authors was supported by the National Natural Science Foundation of China (Grant No. 61272251) and National Key Basic Research Program of China (Grant No. 2015CB856004).

7. REFERENCES

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

[2] J. Canny. A computational approach to edge detection. *IEEE PAMI*, 1986.

[3] Y. Cao, C. Wang, L. Zhang, and L. Zhang. Edgel index for large-scale sketch-based image search. CVPR, 2011.

[4] Y. Cao, H. Wang, C. Wang, Z. Li, L. Zhang, and L. Zhang. Mindfinder: Interactive sketch-based image search on millions of images. MM, 2010.

[5] M. Eitz, K. Hildebrand, T. Boubekeur, and M. Alexa. A descriptor for large scale image retrieval based on sketched feature lines. SBIM, 2009.

[6] X. Sun, C. Wang, C. Xu, and L. Zhang. Indexing billions of images for sketch-based retrieval. MM, 2013.

[7] K.-Y. Tseng, Y.-L. Lin, Y.-H. Chen, and W. H. Hsu. Sketch-based image retrieval on mobile devices using compact hash bits. MM, 2012.

[8] R. Zhou, L. Chen, and L. Zhang. Sketch-based image retrieval on a large scale database. MM, 2012.