Indexing Billions of Images for Sketch-based Retrieval

Xinghai Sun[†,*], Changhu Wang[‡], Chao Xu[†], Lei Zhang[‡]
[†]Key Laboratory of Machine Perception (Ministry of Education), Peking Univ., Beijing, P. R. China
[‡]Microsoft Research Asia, Beijing, P. R. China
{sunxh, xuchao}@cis.pku.edu.cn, {chw, leizhang}@microsoft.com

ABSTRACT

Because of the popularity of touch-screen devices, it has become a highly desirable feature to retrieve images from a huge repository by matching with a hand-drawn sketch. Although searching images via keywords or an example image has been successfully launched in some commercial search engines of billions of images, it is still very challenging for both academia and industry to develop a sketch-based image retrieval system on a billion-level database. In this work, we systematically study this problem and try to build a system to support query-by-sketch for two billion images. The raw edge pixel and Chamfer matching are selected as the basic representation and matching in this system, owning to the superior performance compared with other methods in extensive experiments. To get a more compact feature and a faster matching, a vector-like Chamfer feature pair is introduced, based on which the complex matching is reformulated as the crossover dot-product of feature pairs. Based on this new formulation, a compact shape code is developed to represent each image/sketch by projecting the Chamfer features to a linear subspace followed by a non-linear source coding. Finally, the multi-probe Kmedoids-LSH is leveraged to index database images, and the compact shape codes are further used for fast reranking. Extensive experiments show the effectiveness of the proposed features and algorithms in building such a sketch-based image search system.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing

General Terms

Algorithm, Design, Experimentation

*This work was performed at Microsoft Research Asia.

MM'13, October 21–25, 2013, Barcelona, Spain.

http://dx.doi.org/10.1145/2502081.2502281.

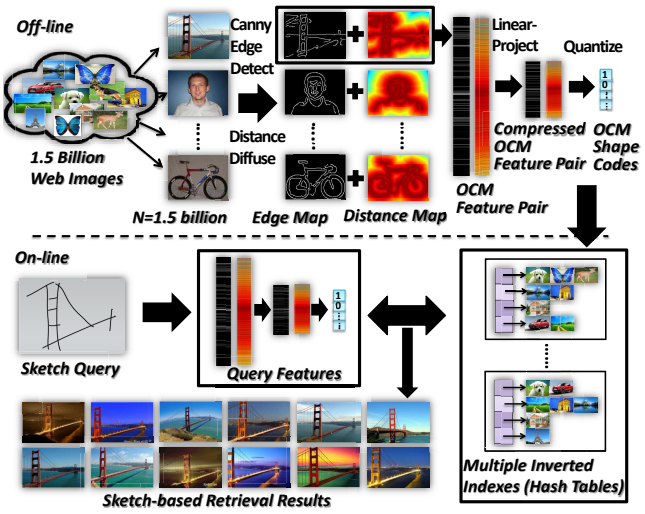


Figure 1: The outline of the sketch-based image retrieval system. Upper part: off-line engine building procedure, including image preprocessing, feature pair extraction, dimension reduction, shape codes generation and index construction. Lower part: the online retrieval.

Keywords

Sketch-based Image Retrieval, Billion Scale, Chamfer Matching, Product Quantization, Indexing

1. INTRODUCTION

Image content analysis has long been a fundamental research problem in the computer vision and multimedia communities. Although much progress has been made in analyzing texture-rich natural images, the understanding of textureless *shape*, which plays an essential role in human vision, is still a challenging problem. Most previous efforts in shape analysis were constrained within small databases, which seriously limit potential discoveries and practical applications. Stepping into the era of big data, the explosive growth of web images motivates us to restudy this problem in a very large scale. We believe a web-scale image data set can provide a great potential to solving many traditionally unsolvable problems. To efficiently utilize billions of images, the first step is to efficiently access the data set. Therefore, in this work, we target at the retrieval problem: to retrieve images similar to a hand-drawn sketch (in terms of shape)

from 1.5 billion images in real time. Based on the success of this capability, other shape/sketch understanding problems, e.g. sketch recognition, shape modeling, shape detection, sketch suggestion, will become possible.

- The Bless of Big Data

We are embracing the era of big data. The exploding amount of web data have revolutionized our daily lives and reshaped the scientific research, including image retrieval and understanding in both academia and industry. Before 2000, most research in multimedia was conducted in thousand-level databases, in which diverse features and sophisticated models were explored. It then moved to its million-era owning to the growth of web images and the development of content-based image retrieval (CBIR). Some challenging problems such as image annotation were studied in a data-driven way[17], and superior performance shows the power of big data. Torralba et al [15] collected 80 million tiny images and showed the great value of data-driven methods in several computer vision tasks. Since 2008, CBIR has moved into a billion-era, and come into commercial use, e.g. query by image examples in TinEye and Google.

- Large-scale Sketch-based Image Retrieval

Finding images by drawing a sketch is a highly desired feature, especially with the popularity of touch-screen devices. Different from keyword-based and example-based methods, sketch-based image retrieval (SBIR) targets at finding images similar to a user's drawing in terms of major shapes. This new retrieval scheme provides a flexible and natural way for users to express their search intentions, which can be combined with other schemes in a complementary way. However, search by sketch is a very challenging task, since the flexibility of the drawing makes the query space very huge. In a small database, it is almost impossible to find an acceptable image if we do not constrain a user's drawing.

As a crucial part of SBIR, shape matching was intensively studied in 1990s in small databases of hundreds to thousands of images, and used in small-scale vertical applications, such as hand-gesture recognition, mould, clip-art and other simple pattern retrieval. To enable an arbitrary drawing as a query, in 2010, we proposed an *edgel-index* structure, based on which a real-time sketch-based search engine *MindFinder*[4, 3] was built on two million natural images. The result was promising, since the system can find similar images for a large portion of simple hand-drawn sketches. However, we also observed that two million images are still far from sufficient to cover the query space. Thus, encouraged by the use of query-by-example technology in commercial search engines such as Google and TinEye, in this work we attempt to advance the state-of-the-art sketch-based image search to the scale of billion-level.

- Algorithm Design

There are two major challenges to build a sketch-based image search engine on a billion-level database: 1) robust shape matching between a hand-drawn sketch and a real image; and 2) efficient index structure and compact representation to support such a large-scale matching. Moreover, these two aspects are intimately coupled with each other, making the problem even more challenging.

To bridge the representation gap, most methods first extract representative contours from real images; some statistical features, e.g. gradient/edge histogram[6], shape context[1], GF-HoG[8], sketch-HoG[7], are further used along with global or bag-of-local-feature representation. Some work designed complex topology models which are however difficult to scale up. Chamfer Matching[2] is another effective way to deal with such a global matching. It measures the similarity of two edge maps by minimizing geometric distance of two edge point sets. To evaluate their effectiveness, we conducted experiments on sketch classification and sketch-based image retrieval to test these representation and matching methods. The experiments showed that Oriented Chamfer Matching[13] (OCM) achieves very competitive performance. Thus, we base our further design on OCM.

Besides shape matching, scalability remains the major issue in a practical system. Most existing sketch-based work can hardly scale up to billion-level databases. The brute-force linear-scan used in small databases is impossible to support such a large-scale task. The edgel-index structure in MindFinder system would require 5TB memory space and cost several minutes for a single query if indexing 1.5 billion images. Other inverted index structure for bag-of-feature (e.g. SHoG) representation also requires more than 3TB memory cost using the recommended number of local features. All of these methods are impractical and unacceptable with only a few common servers. Moreover, Oriented Chamfer Matching is much more time-consuming than L-2 global feature matching, due to the minimization process[1], which makes the problem more challenging.

To make OCM more efficient, we proposed a compact feature representation, whose on-line matching is tens of thousands of times faster than OCM (with on-line DT). The OCM distance is first transformed into the dot-product of two high-dimensional vectors (edge-map vector and distance transform (DT) vector), which is a novel way to look at OCM, as shown in Fig.1. Since DT vector is intrinsically low-dimensional, we linearly project the two vectors onto this low-dimensional subspace without much information loss in dot-product calculation. Product quantization[9] are further utilized to reduce the representation. Finally, each image is represented by 288 bytes, possible to be stored in memory and used for fast near-accurate OCM computation.

Based on this representation, an efficient index structure is built. We turn to hashing techniques in large-scale approximate nearest neighbor search. Since the unstructured k-means hash[11] outperforms many structured hash methods, we utilize k-medoids hashing for fast but only coarse indexing (Fig.1). By leveraging this hashing structure, a relatively small portion of OCM distances need to be computed for reranking. The reranking stage is very efficient, since based on the compact representation, we only need to calculate the dot-product between two short vectors. It could support 2×10^6 OCM computations within a second in one common server. Based on this design, a system was built using 10 common servers, each of which indexes $100 \sim 200$ million images with approximately 60GB memory cost, supporting real-time response. The outline of this system is shown in Fig.1. Intensive experiments have shown the effectiveness of the proposed features and methods in building a billion-scale sketch-based retrieval system.

The rest of the paper is organized as follows: Section 2 introduces the compact feature representation and matching approach. The index structure is presented in Section 3. Po-

[1]A distance-transform(DT) map could be constructed, which actually uses memory storage to reduce time cost, but is still impractical for billions of images.

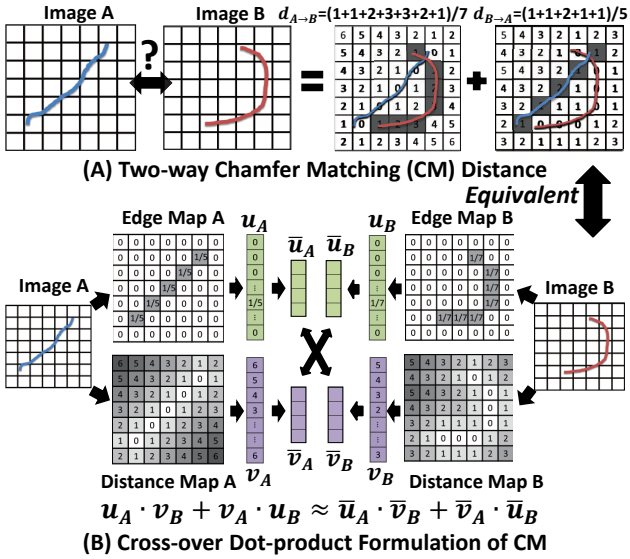


Figure 2: The crossover dot-product reformulation for Chamfer matching. (a) Two-way Chamfer matching (CM). (b) Crossover dot-product reformulation of CM. Each image is represented by a Chamfer feature pair: edge-map vector u and distance-map vector v. The classic two-way Chamfer matching distance is equivalent to the crossover dot-product of the feature pairs, facilitating further dimension compression ($\bar{u}$ and $\bar{v}$) and compact shape codes generation.

tential applications and extensive experiments are provided in Section 4 and 5, followed by the conclusion in Section 6.

2. COMPACT SHAPE CODING AND MATCHING

In this section, we first introduce the basic feature representation and matching approach adopted in this work, i.e. Oriented Chamfer Matching (OCM)[13] for edge maps. Then, a new reformulation for OCM is proposed, based on which a compact representation and matching is introduced. Finally, we present the multi-layer source coding for OCM to further compress the representation.

2.1 Feature Representation and Matching

Developing an effective shape feature and its corresponding matching method is a fundamental and preliminary problem to any shape-related tasks. Intensive research efforts have been spent on this problem, including local and global features based on gradient/edge histogram[6], shape context[1], sketch-HoG[7], and corresponding *bag-of-visual-word* representations to organize local features. Different features will correspond to different matching methods, such as histogram intersection for bag-of-visual-words, and Chamfer Matching[2, 3] for raw edge matching.

In this work, to choose an appropriate feature and matching method as a basic representation, we conducted experiments on sketch classification and sketch-based image retrieval to evaluate these representation and matching methods. Experiment results (in Sec.5.1) show that OCM achieves very competitive performance in these two sketch-related tasks. Thus we build our retrieval algorithm and system on OCM, which will be briefly introduced below.

Oriented Chamfer Matching

Chamfer matching[2] is a basic shape matching method proposed in the early 1990s. Different from the statistical information in histogram-like features, Chamfer matching bases its matching process on pixel-level: it measures the geometric distances of two edge-pixel sets from two images. Let $\mathcal{I}_\mathcal{A} = \{\mathbf{x}_a\}$ and $\mathcal{I}_\mathcal{B} = \{\mathbf{x}_b\}$ denote the edge pixel sets of images A and B, where $\mathbf{x}_a = (x_a, y_a)$ is the geometric coordinates of edge pixels. The one-way Chamfer distance from $\mathcal{I}_\mathcal{A}$ to $\mathcal{I}_\mathcal{B}$ is defined as:

$$D^{CM}_{A\to B} = \frac{1}{|\mathcal{I}_\mathcal{A}|} \sum_{\mathbf{x}_a \in \mathcal{I}_\mathcal{A}} \min_{\mathbf{x}_b \in \mathcal{I}_\mathcal{B}} \parallel \mathbf{x}_a - \mathbf{x}_b \parallel_2, \qquad (1)$$

and the final two-way Chamfer Distance is given by:

$$D^{CM}_{A,B} = \frac{1}{2}(D^{CM}_{A\to B} + D^{CM}_{B\to A}). \qquad (2)$$

To encode the orientation information of curves at edge pixels, Oriented Chamfer Matching (OCM)[13] was proposed to quantize the orientation of edge pixels into different channels and perform the basic Chamfer matching on each channel. Assume each edge set is represented by $|{\mathcal{I}_\mathcal{A}}^\theta|$ sub-sets with different orientations: $\mathcal{I}_\mathcal{A} = \{{\mathcal{I}_\mathcal{A}}^\theta \mid \theta \in \Theta\} = \{\{\mathbf{x}^\theta_a\} \mid \theta \in \Theta\}$, where Θ is a quantized orientation set, ${\mathcal{I}_\mathcal{A}}^\theta$ is a subset with edge pixels $\mathbf{x}^\theta_a$ of quantized orientation $\theta \in \Theta$. Oriented Chamfer Matching is defined by:

$$D^{OCM}_{A\to B} = \frac{1}{|\mathcal{I}_\mathcal{A}|} \sum_{\theta\in\Theta} \sum_{\mathbf{x}^\theta_a \in {\mathcal{I}_\mathcal{A}}^\theta} \min_{\mathbf{x}^\theta_b \in \mathcal{I}_\mathcal{B}} \parallel \mathbf{x}^\theta_a - \mathbf{x}^\theta_b \parallel_2, \qquad (3)$$

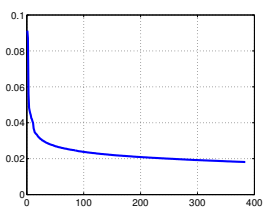
with its two-way form similar to Eqn.2. In this work, the edge pixel's orientation is quantized into 6 bins, i.e. $-15^\circ \sim 15^\circ, 15^\circ \sim 45^\circ, \ldots, 135^\circ \sim 165^\circ$.

2.2 Compact Representation for OCM

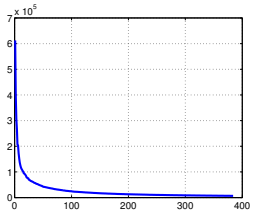
In spite of good matching performance, OCM's matching function requires intensive computation and large memory cost. By leveraging an edge-pixel-based inverted index structure (Edgel Index) to speed up searching, we have built a real-time sketch-based image search engine[4, 3] based on Chamfer matching on a two-million image database. However, to save the memory cost, in [3] the two-way OCM was simplified to a one-way OCM search followed by a two-way OCM reranking for the top 5000 resulting images. However, when the database is enlarged by 750 times to 1.5 billion, this reranking scheme will not work since the recall will be quite low using one-way OCM search. On the other hand, even the one-way OCM search cannot work here since the whole index will cost 5TB memory, which is unaffordable for common servers. It should be noted that it is also impractical to only store the coordinates of edge pixels of each image, since on the one hand, it will also take 3TB memory space (assuming 1000 edge pixels per image), and more importantly, it is intractably inefficient to generate the distance-maps on-line for all the database images. Thus, a more compact representation for raw edges and their efficient OCM are highly desired.

Crossover Dot-Product Reformulation for OCM

Distance transformation (DT) is often used to speed up the on-line Chamfer distance computation. Assuming to



(a) Edge-map Vector $\mathbf{u}$



(b) Distance-map Vector $\mathbf{v}$

Figure 3: Spectrums of a) edge-map vector u: intrinsically high-dimensional, and b) distance-map vector v: intrinsically low-dimensional.

calculate $D^{OCM}_{A\to B}$, we first compute a distance-map $DT^{\theta}_{\mathcal{I}_\mathcal{B}}$ for each channel of $\mathcal{I}_\mathcal{B} = \{\mathcal{I}_\mathcal{B}{}^1, \mathcal{I}_\mathcal{B}{}^2, \ldots, \mathcal{I}_\mathcal{B}{}^\theta\}$. The distance-map value for each geometric location $\mathbf{x}$ measures the minimal distance from this location to an edge in $\mathcal{I}_\mathcal{B}{}^\theta$:

$$DT^{\theta}_{\mathcal{I}_\mathcal{B}}(\mathbf{x}) = \min_{\mathbf{x}^{\theta}_b \in \mathcal{I}_\mathcal{B}{}^\theta} \parallel \mathbf{x} - \mathbf{x}^{\theta}_b \parallel_2, \tag{4}$$

Then Eqn. 3 is equal to:

$$D^{OCM}_{A\to B} = \frac{1}{|\mathcal{I}_\mathcal{A}|} \sum_{\theta\in\Theta} \sum_{\mathbf{x}^{\theta}_a \in \mathcal{I}_\mathcal{A}{}^\theta} DT^{\theta}_{\mathcal{I}_\mathcal{B}}(\mathbf{x}^{\theta}_a). \tag{5}$$

When $DT(\cdot)$ is computed off-line, the complexity of online matching could be reduced from $\mathcal{O}(|\mathcal{Q}|\times|\mathcal{D}|)$ to $\mathcal{O}(|\mathcal{Q}|)$, with the cost of increased storage of distance-maps. Assuming the size of an image is 160×160 and the orientation is quantized into 6 bins, the distance maps for each image have to be stored with 0.6 MB ($160\times160\times6\times4$ bytes) and for 1.5 billion images it will cost 900 TB, which is difficult to be stored on disks of common servers, let alone to load them in memory for fast access.

To develop a more compact representation and speed up the matching, we first convert a sketch/image to a vector representation, based on which the OCM can be conducted by the dot-product of vectors.

Therefore, two vectors are defined: 1) **edge-map vector** (denoted as $\mathbf{u}$), by concatenating edge-map matrixes ED^θ of all the orientation channels, and 2) **distance-map vector** (denoted as $\mathbf{v}$), by concatenating distance-map matrixes DT^θ of all the $|\Theta|$ orientation channels:

$$ED^\theta = [e_{ij}],\ e_{ij} = \begin{cases} 1/|\mathcal{I}|, & \text{if } \mathbf{x}^{\theta}_{ij} \in \mathcal{I}^\theta; \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

$$\mathbf{u}^\theta = \text{vec}(ED^\theta),\quad \mathbf{u} = [\mathbf{u}^1, \mathbf{u}^2, \ldots, \mathbf{u}^\theta]^{\mathrm{T}}, \tag{7}$$

$$\mathbf{v}^\theta = \text{vec}(DT^\theta),\quad \mathbf{v} = [\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^\theta]^{\mathrm{T}}, \tag{8}$$

where $\text{vec}(\cdot)$ denotes the vectorization operation on matrix. Both $\mathbf{u}$ and $\mathbf{v}$ are of $153,600$ ($160\times160\times6$) dimensions, with $\mathbf{u}$ being sparse and $\mathbf{v}$ being dense.

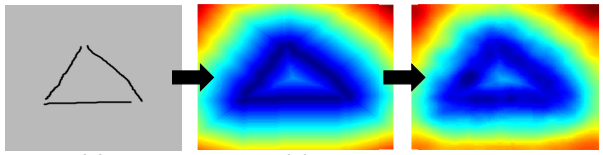
Then, the one-way OCM can be simply written as the dot-product of these two vectors:

$$D^{OCM}_{A\to B} = \mathbf{u}^{\mathrm{T}}_B \mathbf{v}_A. \tag{9}$$

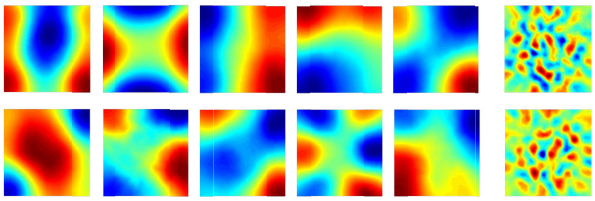
The two-way (symmetric) OCM is given by:

$$D^{OCM}_{A,B} = \frac{1}{2}(\mathbf{u}^{\mathrm{T}}_B \mathbf{v}_A + \mathbf{v}^{\mathrm{T}}_B \mathbf{u}_A), \tag{10}$$

where $(\mathbf{u}_A, \mathbf{v}_A)$ and $(\mathbf{u}_B, \mathbf{v}_B)$ are Chamfer feature pairs of $\mathcal{I}_\mathcal{A}$ and $\mathcal{I}_\mathcal{B}$. The two-way OCM is taken as the crossover dot-product form of the Chamfer feature pairs, as illustrated in Fig.2.



(a) Distance-map Reconstruction



(b) Bases for Reconstruction

Figure 4: Illustration for distance-map reconstruction with principle component bases. (a) From left to right, edge map of an example sketch, original distance-map, reconstructed distance map with 256 principle components. (b) Eigen-maps of distance transform maps. The left five columns show the top ten principle components of the distance-transform map, serving as some of the reconstruction bases for compressing the distance-transform map. The last column shows the 300th and 301st components of the distance-transform map, containing high-frequency information.

The crossover dot-product formulation of Oriented Chamfer distance is a new way to look at the classic OCM, making it easier for further compression and analysis.

Linear Subspace Projection

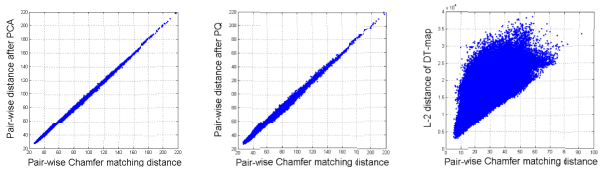
It is impractical to directly store and compute $\mathbf{u}$ and $\mathbf{v}$ in large-scale tasks, since both vectors are high-dimensional. We perform PCA analysis on these two vectors, and found that $\mathbf{v}$ is intrinsically low-dimensional while $\mathbf{u}$ is not. The sorted eigenvalues of the covariance matrix of sampled $\mathbf{u}$ and $\mathbf{v}$ vectors are shown in Fig.3(a) and Fig.3(b). It can be seen that the distance-map vector $\mathbf{v}$ are intrinsically distributed in a linear subspace of much lower dimension: only about two hundred principal axes can cover most variations in the original feature space. Notice that the Chamfer matching distance is already represented as crossover dot-product of $\mathbf{u}$ and $\mathbf{v}$ (in Eqn.9). Although $\mathbf{u}$ is not intrinsically low-dimensional, we can preserve the dot-product between $\mathbf{u}$ and $\mathbf{v}$, by projecting them onto $\mathbf{v}$'s subspace.

Let us denote $U_{D\times d}$ as the orthogonal projection matrix learned by PCA analysis for sampled vectors $\mathbf{v}$ (D is the original dimension number and d is the reduced dimension number, $d \ll D$). We project both $\mathbf{u}$ and $\mathbf{v}$ to the subspace using $U_{D\times d}$ to generate a compact representation $\bar{\mathbf{u}}$ and $\bar{\mathbf{v}}$:

$$\bar{\mathbf{u}}_{d\times1} = U^T_{D\times d}\mathbf{u}_{D\times1}, \tag{11}$$

$$\bar{\mathbf{v}}_{d\times1} = U^T_{D\times d}\mathbf{v}_{D\times1}. \tag{12}$$

We now show that the dot-product between $\mathbf{v}$ and $\mathbf{u}$ is preserved under the new representation. $\mathbf{v}$ and $\mathbf{u}$ can be



(a) PCA-reduced (b) PQ-Coding (c) L2 of DT-map

Figure 5: Pair-wise distances comparison. We sampled one million pairs of images and compared different distances against the groundtruth OCM distances. The X-axis represents the OCM distances, while the Y-axis represents: a) OCM after the PCA dimension reduction, b) OCM after product quantization coding (in Sec.2.3), and c) the L-2 distances of distance-transform (DT) map[16]. Figure (a) and (b) show good linear correlations with very minor distance distortion; Figure (c) have very big distance distortions, showing that the L-2 distances of DT-map is inconsistent with Chamfer matching.

written as below:

$$\mathbf{u} = U\bar{\mathbf{u}} + \mathbf{e}_u, \tag{13}$$

$$\mathbf{v} = U\bar{\mathbf{v}} + \mathbf{e}_v, \tag{14}$$

where $\mathbf{e}_v$ is a residual vector, which is very small. Although $\mathbf{e}_u$ is not small, it is orthogonal to the projected subspace. Thus we have,

$$\begin{aligned} \mathbf{u}^T\mathbf{v} &= (\bar{\mathbf{u}}^T U^T + \mathbf{e}_u^T)(U\bar{\mathbf{v}} + \mathbf{e}_v) && (15)\\ &= \bar{\mathbf{u}}^T U^T U\bar{\mathbf{v}} + (\mathbf{e}_u^T U)\bar{\mathbf{v}} + \bar{\mathbf{u}}^T(U^T\mathbf{e}_v) + \mathbf{e}_u^T\mathbf{e}_v && (16)\\ &= \bar{\mathbf{u}}^T\bar{\mathbf{v}} + \mathbf{e}_u^T\mathbf{e}_v && (17) \end{aligned}$$

Since $|\mathbf{e}_u^T\mathbf{e}_v| \le |\mathbf{e}_u||\mathbf{e}_v| \le |\mathbf{u}||\mathbf{e}_v| \le |\mathbf{e}_v|$ and $|\mathbf{e}_v|$ is negligible, the dot-product of two high-dimensional features $\mathbf{u}$ and $\mathbf{v}$ are well preserved under the dot-product of new low-dimensional features $\bar{\mathbf{u}}$ and $\bar{\mathbf{v}}$.

Now we have a more compact representation for the Chamfer distance between a sketch and an image using their vector-based representation:

$$D_{A,B}^{OCM} = \frac{1}{2}(\bar{\mathbf{u}}_B^{\mathrm{T}}\bar{\mathbf{v}}_A + \bar{\mathbf{v}}_B^{\mathrm{T}}\bar{\mathbf{u}}_A), \tag{18}$$

with $\bar{\mathbf{u}}$ and $\bar{\mathbf{v}}$ are only d-dimensional ($d \ll D$). d is typically set to be 256. The distance distortion of the compression is illustrated in Fig.5, from which we can see that the sampled pair-wise distances are mostly preserved when the vector dimensions are reduced from 153,600 to 256, amounting to only 0.17%. Quantitative analysis is provided in Sec.5.2.

Such a compression is actually like a low-pass filter. The top 10 linear principle components of the distance-map are illustrated in the left part of Fig.4(b), while the 300th and 301st principle components are in the right part. It can be seen that top principle components usually capture the informative low-frequency information of the distance map, while the remaining components contain more noisy high-frequency part. After truncating the high-frequency component tails, these principle ***eigen-maps*** are adequate to build a group of bases to reconstruct original distance maps, as shown in Fig.4(a).

This dot-product reformulation and the compression technique greatly simplify the classic Chamfer matching in both computation and memory cost. It essentially converts the intensive on-line computation (e.g. seeking the closest edge point) to off-line (e.g. generating distance-transform map and projecting) while cutting down the feature dimension. Currently, each Chamfer matching only requires several hundred on-line multiplication, much faster than the original $\mathcal{O}(|\mathcal{Q}| \times |\mathcal{D}|)$ geometric distance calculation.

2.3 Non-linear Multi-layer Source Coding

For each image, the low dimensional features $\{\bar{\mathbf{u}}, \bar{\mathbf{v}}\}$ (e.g. 256 dimensional) still require a huge memory cost for a large database. For example, for 1.5 billion images, it will cost 3TB memory space with "float" type, which cannot be stored in memory and have to be read from disk on-the-fly. However, the relatively low speed of random disk accessing severely limits the number of images that could be online ranked (or re-ranked), and in that case an accurate indexing structure is needed to avoid large number of reranking.

Some recent approximate nearest-neighbor (ANN) search techniques inspired by source coding goes another way, such as product quantization[9], which is proved to achieve state-of-the-art ANN performance and outperforms LSH in terms of the trade-off between memory usage and accuracy[9]. Product quantization focuses on explicitly approximating vectors using short codes, so that a large number of compressed codes can be loaded into memory to support a very large portion of reranking after only a coarse indexing. We utilize this technique to further quantize both $\bar{\mathbf{u}}$ and $\bar{\mathbf{v}}$ vectors of database images, and encode them to a more compact code, which can be stored in memory for fast reranking for a large number of images obtained by a coarse index (Sec.3).

In this section, we will first introduce product quantization for OCM, followed by the multi-stage residue coding.

Product Quantization

Product quantization[9] decomposes the vector space into a Cartesian product of low-dimensional subspace, and quantizes each subspace separately, which achieves a good balance between compression rate and learning difficulty.

We briefly introduce the product quantization[9] below. Denote $x \in \mathcal{R}^d$ as the vector to be quantized. x is first split into m subvectors[2] $x^1, \ldots, x^m \in \mathcal{R}^{d/m}$. For each subvector x^i, a sub-quantizer $q^i(\cdot)$ is learned by k-means, mapping the real-value subvector x^i to one of the k_s quantization centroids $c_i = q^i(x^i)$. Then the global product quantizer is:

$$q(x) = (q^1(x^1), q^2(x^2), \ldots, q^m(x^m)). \tag{19}$$

We utilize product quantization to generate compact codes for Chamfer features. In the off-line stage, each image is encoded by a vector of sub-quantizer's IDs:

$$s(x) = (s^1(q^1), s^2(q^2), \ldots, s^m(q^m)), \tag{20}$$

in which $s^i(q^i) \in [0, k_s - 1]$, is the ID of the quantization centroid $q^i(x^i)$.

Although k_s is typically set to a small value (e.g. 256 or 4096) for fast learning and quantization, the number of possible values of $q(x)$ is large, i.e. $(k_s)^m$, since the centroid codebook of the whole vector is the Cartesian product of the centroids of the m sub-quantizers. Such a large number of quantization centroids provide explicit approximation of the original vectors. The centroid index value for each vector

[2]In practice, vectors are multiplied by a random orthogonal matrix prior to quantization to balance the energy of each subvector. Query vectors are also multiplied by the matrix.

could be encoded with $\lceil m \log_2(k_s) \rceil$ bits. Thus, memory space is saved a lot.

Here **Asymmetric Distance Computation** (ADC)[9] is adopted: the database image feature x is encoded to be $s(x)$, but the query feature y is not encoded. Then, the distance $d^{cm}(x, y)$ is approximated by $d^{cm}(q(x), y)$. Thus, there is no approximation error in the query side.

In the on-line retrieval, $d^{cm}(q(x), y)$ could be computed using the decomposition:

$$d^{cm}(q(x), y) = q(x)^T y = \sum_{j=1,\ldots,m} q^j(x^j)^T y^j, \quad (21)$$

where y^j is the j^{th} subvector of the query y. For each j, we calculate the dot-product between y^j to k_s quantization centroids of $q^j(\cdot)$ to construct a look-up table $t(1), ..., t(k_s)$. Then, for each database image, we check its shape code $s(x)$, and sum up all the corresponding values in the look-up table as the final distance to the query y:

$$d^{cm}(q(x), y) = \sum_{j=1,\ldots,m} t(s^i). \quad (22)$$

Multi-stage Residue Coding

Inspired by the work of [9, 10], we refine the first-stage quantization by re-coding their residues. Since the residues often contain lower energy/information than raw vectors, encoding the residue is easier than encoding the original data vector. The residue vector after the quantization is:

$$r(x) = x - q(x), \quad (23)$$

which is further quantized by another quantizer q_r learned on a sampled set of residue vectors. Thus, an improved estimation $\hat{x}$ of x is the sum of approximated vector and the encoded residue vector:

$$\hat{x} = q(x) + q_r(r(x)). \quad (24)$$

Notice that increasing the centroid number k_s of each sub-quantizer can also improve the quantization accuracy. However, a larger k_s will severely increase the computation burden in both learning and quantization processes. With residue encoding, the quantization error is greatly reduced while the computation cost is only linearly increased.

In this work, we perform two-stage residue encoding. At each stage, we set k_s to 4096, and jointly quantize every eight dimensions, which achieves a good balance among compression rate, accuracy and computation cost. Detailed experiments are shown in Sec.5.2.

Finally, the Chamfer features $\bar{\mathbf{u}}$ and $\bar{\mathbf{v}}$ are encoded with $3 \times \log_2(4096) \times 256/8$ bits, and thus each image costs 288 bytes. This feature will be used in the fast reranking stage.

3. MULTI-PROBE KLSH FOR CHAMFER INDEXING

Although the proposed compact shape codes can be stored in memory and support 2×10^6 shape matchings per second on a common server, for a real-time search task of 1.5 billion images, an effective indexing structure is still necessary to prune the obviously unrelated images.

In this work, we build Kmeans-LSH (KLSH)[11] index[3] for images with the dissimilarity defined as the Chamfer distance in Eqn.18. Although Chamfer distance is actually not a metric (does not satisfy the triangle inequality), the experiments showed the effectiveness of KLSH in our system. Details are provided in Sec.5.3. Below we will briefly introduce KLSH, followed by implementation details.

Multi-Probe KLSH

KLSH utilizes unstructured k-means quantizers as hash functions:

$$h_i(x) = \arg \min_{j=1,\ldots,k} d(x, c_j^i), \quad i = 1, \ldots, l. \quad (25)$$

where k is the cluster number of the k-means quantizer and also the number of hash buckets; $d(\cdot, \cdot)$ is the dissimilarity metric; c_j is the cluster centroids. Each hash value $h_i(\cdot)$ represents a hash bucket and all database objects are put into one of the buckets according to their hash values. Thus, each h_i function forms a hash table, and the objects falling in the same bucket will have a larger likelihood of being nearest neighbors than those disseminated in different buckets. Then, multiple (l) hash tables/functions (with different k-means initiates) are constructed to reduce the probability that a vector is missed, as in the LSH framework. When a user draws a query sketch and triggers the search, the hash value of query object y is obtained by $h_i(y)$, and the objects in the bucket of the value $h_i(y)$ in each hash table are returned as the resulting short-list for further reranking.

Since the memory cost increases with the number of hash tables[4] l, to reduces memory usage, multi-probe KLSH[11] is used here to retrieve several (m_p) buckets per hash table, instead of only one. Then for each query, the total number of buckets retrieved becomes $l \times m_p$. Therefore, for a fixed number of buckets, the number of hash tables is reduced by a factor of m_p, so is the memory cost.

Implementation Details

We built the sketch-based image retrieval system with 10 common servers to support a realtime search on 1.5 billion images. Each server indexes $100 \sim 200$ million images with about 60GB memory cost. The memory cost includes two parts: KLSH index and the shape codes. For the hash index, we constructed $l = 5$ hash tables each of which consists of $k = 8192$ hash buckets and the total index will cost 4GB. For the shape codes part, we truncated both u and v to be of 256 dimensions, and every eight dimensions were jointly quantized and encoded with 4.5 ($= 3 \times 1.5$) bytes. The total memory cost for 200 million images is 54GB. Given a query, the hash index in each server returns a list of 2×10^6 images, which are further reranked with Chamfer shape codes. The response time for each query is $0.5 \sim 1.5$ seconds.

4. POTENTIAL APPLICATIONS

Based on this billion-scale sketch-based image retrieval system, many applications which were thought very challenging or even impossible might become easier and feasible.

1. Sketch Recognition (Fig.6(a)) In the era of touch screens, automatically recognizing a hand-drawn sketch has become an important research problem, and will benefit many touch-related applications. Most existing methods mainly focus on recognizing simple shapes in specific domains, or classifying sketches to a limited number of cate-

[3]The k-medoids clustering was used instead of k-means clustering, since we have to deal with a space only defined by dissimilarity.

[4]l hash tables will cost $n \times l \times 4$ bytes, where n is the number of indexed objects. When $n = 1.5 \times 10^9$, each single hash table will cost 6GB memory.

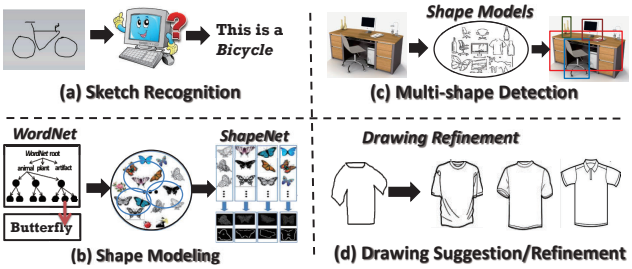
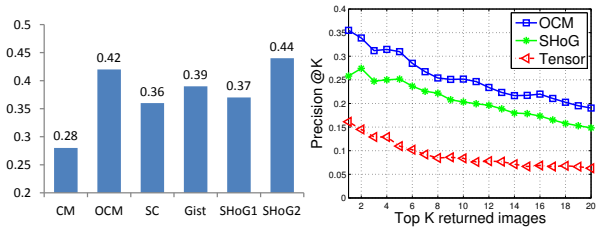


Figure 6: Four potential applications that could be advanced by the retrieval system.



(a) Sketch Classification (b) Image Retrieval

Figure 7: Results of different shape features on sketch classification and sketch-based image retrieval tasks. (a) Sketch classification accuracy of CM[2], OCM[13], SHoG[5], Shape Context (SC)[1], and Gist[12]. (b) Sketch-based image retrieval performance (Precision@K) of OCM[13], SHoG[7] and Tensor[6].

gories. To recognize an arbitrary drawing, we have developed a data-driven method[14] based on one million clipart images. However, million-scale images are still far from sufficient to cover the query space. We believe that with the help of 1.5 billion images and the sketch retrieval techniques, sketch recognition will become more practical.

2. Shape Modeling (Fig.6(b)) Automatically mining and modeling shapes for an arbitrary concept is a fundamental problem in visual concept modeling in computer vision. Most existing work only focuses on modeling a very limited number of objects. Based on this system and its backend index structure, it becomes possible to model any concept with meaningful shapes, which will be useful to many computer vision problems.

3. Multi-Shape Detection (Fig.6(c)) Object detection is another important problem in computer vision. Most existing work focuses on detecting a single object, such as face, human body, and car. Once we collect a large number of shape models, it might be possible to conduct multi-shape or multi-object detection. For example, if we have typical shape models for vehicles, persons, houses, trees, etc., it will be helpful for detecting related objects in a natural image by combining with other features.

4. Drawing Suggestion & Refinement (Fig.6(d)) Drawings from non-experts are usually unartistic, and sometimes even ugly. By recognizing a user's partial sketch and mining possible shape intentions, the system can suggest better drawings to help the user quickly get an elegant drawing, which is useful for kids education and quick interaction.

5. EXPERIMENTS

Extensive experiments were conducted to evaluate the entire system as well as different components including feature selection, dimension reduction, product quantization, and indexing structure. We explore the impact of data scale in sketch-based retrieval, showing the promising advantages coming along with the increasing growth of data scale.

5.1 Feature Comparison

Several representative shape features were evaluated on two different sketch-related tasks, i.e. sketch classification and sketch-based image retrieval.

Sketch classification was conducted on the 20K hand-drawn sketch data set[5] (20K sketches in 250 categories), and five representative features for shape matching were compared, i.e. CM[2], OCM[13], Shape Context[1], Gist[12] and SHoG[5] [5] (SHoG1 learns the codebook only on the training set, while SHoG2 learns it on the whole data set as in [5]). We used 3-fold cross-test: the data set was randomly partitioned into three parts, in which one was used as testing set and the remaining two as training set. The accuracies averaged over all three folds are reported in Fig.7(a).

Sketch-based image retrieval was conducted on the 102K image dataset with 31 hand-drawn sketches[7], on which SHoG was proved outperforming other local features in [7]. We compared OCM[13], SHoG[7], and Tensor[6], all of which were successfully leveraged in sketch-based image retrieval systems. The Precision@K curves are shown in Fig.7(b).

The experimental results showed the competitiveness of OCM compared with other features in these two tasks, and thus we base this system on OCM.

5.2 Compression Loss Analysis

In this section, we evaluate the loss of both compression components of PCA dimension reduction and product quantization under different parameter configurations.

Metric for Compression Loss

We use the following metrics to evaluate the compression loss:

1) **Root Mean Square Error (RMSE)**: is defined as $\sqrt{\sum_i (d_i - \hat{d}_i)^2}$, where d_i is the OCM distance and $\hat{d}_i$ is the approximated distance after compression. 10^7 pairs of images were randomly sampled for this test.

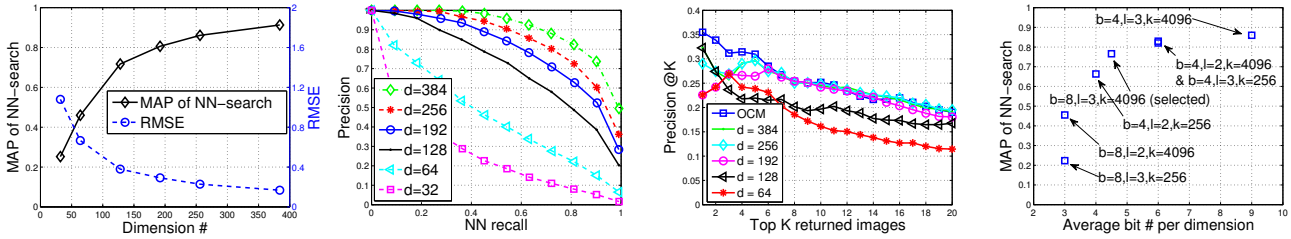
2) **Precision-Recall (PR) & MAP for NN-search**: based on nearest-neighbor search, we consider images within an OCM proximity (k-NN, e.g. k=100) of each query as "positive" while the rest "negative". By recalling images based on the approximated distances, the precision-recall curves (PR) and mean average precision (MAP) for the approximated distance can be plotted.

3) **Precision @K for SBIR**: Retrieval Precision @K directly evaluates how the compression procedure impacts the final retrieval performance. We perform this experiment in the same 102K dataset with 31 queries as aforementioned in Sec.5.1. The top 20 results were labeled as "related" or "unrelated" to the query sketch in terms of shape, based on which Precision@K is evaluated for retrieval performance.

PCA Dimension Reduction

As explained in Sec.2.2, the dimension of Chamfer features is reduced by projecting $\mathbf{u}$ and $\mathbf{v}$ onto a subspace of

[5]The local feature in [5] is named as SHoG here, which is a little different from that in [7].



(a) RMSE & MAP of PCA (b) PR-Curves of PCA (c) Prec@K of PCA (d) MAP of PQ

Figure 8: Compression loss analysis. (a-c) The compression loss for PCA dimension reduction ($d = 32, 64, 128, 256, 384$ for $\bar{\mathbf{u}}$ and $\bar{\mathbf{v}}$), evaluated by RMSE, MAP, precision-recall curves of nearest-neighbor search, Precision@K curves of real retrieval performance; (d) The loss in product-quantization coding, evaluated by MAP of NN-search, under different parameter configurations about joint quantization dimension number b_s, encoding layers n_l and number of quantization centroids k_s.

$\mathbf{v}$ to preserve their dot-product value $\mathbf{u}^T\mathbf{v}$. Small distance distortion might occur during the reduction. With the aforementioned metrics, we evaluated how the Chamfer distance distorts when vectors are reduced to different dimensions. Fig.8(a-c) show the results under the metric of RMSE, PR curves, and Precision@K with different vector dimensions. Note that vectors of more than 256 dimensions no longer lead to significant accuracy improvements. Thus, the dimension of $\bar{\mathbf{u}}$ and $\bar{\mathbf{v}}$ is set to 256 to achieve a reasonable balance between memory cost and approximation accuracy.

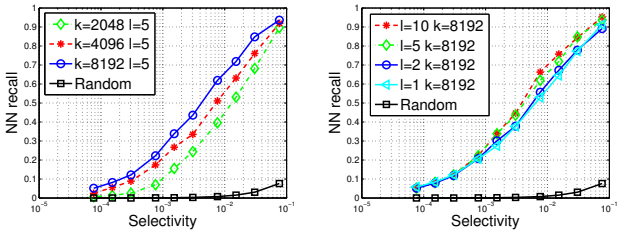
Product Quantization Coding

$\bar{\mathbf{u}}$ and $\bar{\mathbf{v}}$ are further encoded into short bits by product quantization as introduced in Sec.2.3. Factors that impact the quantization performance include the dimension number for joint quantization b_s, the quantization centroid number k_s, and the encoding layer number n_l. Large k_s and n_l with small b_s will increase the actual allocated bit number per dimension, and thus reduce the quantization error at the cost of more total bits; a large k_s helps improve the quantization performance at the cost of increasing computation cost during quantizer learning and coding process. We evaluated several parameter configurations, as shown in Fig.8(d). To balance the memory cost, accuracy, and computation cost, we choose $b_s = 8$, $n_l = 3$, and $k_s = 4096$.

5.3 Impact of KLSH

In this section, we measure the quality of the k-medoids hashing structure in terms of *selectivity-recall curve*[11]. *Selectivity* represents the ratio of the total selected images (for further reranking) among the indexed corpus. The *recall* measures the ratio of recalled k-NN images among k. Here k is set to 200. The experiment was performed on the introduced hash structure which indexes 200 million images[6].

The index quality is mainly related to two factors: the number of hash buckets k and the number of hash tables l. The selectivity-recall curves with $l = 5$ and different k are shown in Fig.9(a), while curves with $k = 8192$ and different l in Fig.9(b). It is observed that a larger k or l will improve the index quality. However, a larger k increases the off-line computation cost for constructing the index, and a large l increases the index's memory cost by a factor of l. It should be noted that, as aforementioned, since the reranking process are significantly accelerated by the proposed shape



(a) Different k (b) Different l

Figure 9: Evaluations of the effectiveness of Kmeans-LSH in Chamfer matching distance space with sensitivity-recall curves at different k and l.

codes, a large portion of images (e.g. 2×10^6 images per second per server) are possible to be reranked. Therefore, we only need a coarse index. We choose $k = 8192$ and $l = 5$ in our system, with which 10^{-2} selectivity can obtain 68% recall of the 200-NN images, at a relatively small cost for off-line index building.

5.4 Evaluations for the Whole System

To evaluate the sketch-based retrieval system, we conducted a series of experiments on a large image corpus consisting of 1.5 billion images. We show how the system performance varies with increasing image set size, showing the necessity of studying billion-scale SBIR. We show that the traditional keyword-based image search results could be significantly enhanced when cooperating with sketch-based search. Besides, we also analyze how the quality of user-drawn sketch queries impacts the system performance.

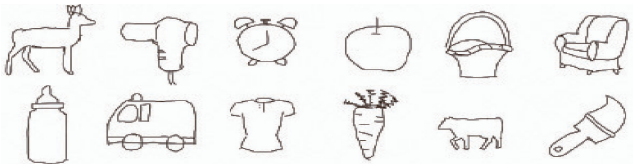
Sketch-500 Query Set* & *1.5-Billion Image Set

The *Sketch500* data set [14] were used as the query set. It consists of 500 hand-drawn sketches (see Fig.10(a)) built based on a list of 1000 commonly-used non-abstract nouns collected for elementary education[7]. These sketches were drawn by a graduate student based on top results returned from a commercial keyword-based image search engine. By removing some non-entity words that are difficult to depict, such as "earthquake", "storm", and "dust", 500 sketches were finally collected together with their keyword texts.

1.5 billion thumbnails randomly sampled from Bing were utilized to build the search system. Some text information

[6]This is because in our system each server indexes less than 200 million images, as introduced in Sec.3

[7]http://www.free-teacher-worksheets.com/support-files/list-of-nouns.eps



(a) Examples in Sketch500 dataset

(b) Illustrations for the strict labeling rule

Figure 10: (a) Examples of hand-drawn sketches in Sketch500 query set. (b) Examples of labeling rule — both shape and concept consistency. the 1st column shows example queries; the 2nd and 3rd columns are images labeled with "correct" since their shapes and concepts are consistent with the queries'; column 4∼7 are "incorrect" due to their incorrect concepts or dissimilar shapes.

such as image titles and surrounding texts together with *tfidf* ranking were leveraged for keyword-based search.

Evaluation Measurement and Labeling Methodology

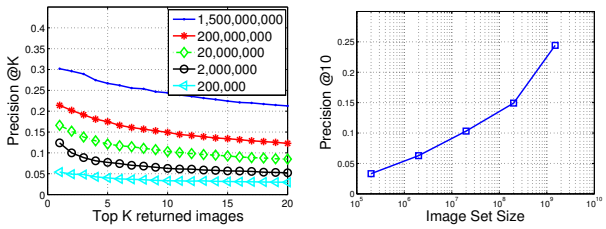
Since it is impossible to manually label all the 1.5 billion images, we adopted a strategy commonly used in [3, 6]: only label the top 20 returned search results for each query and measure the retrieval performance with Precision@K (the ratio of the "correct" images in top K returned images). This is consistent with the observed fact that users usually care about the top results returned by a search engine. For labeling principles, we set a strict rule: only images that meet the following two requirements were labeled as "correct": 1) of similar shape to the query sketch and 2) of consistent concept with the query. As illustrated in Fig.10(b), images with either obvious shape dissimilarity or concept inconsistency were labeled as "incorrect". Note that, if the requirement of concept consistence is relaxed, the retrieval performance will be much higher than what appears in our experiments.

Performance on Different Image Set Scale

A series of experiments were conducted on 5 different scales of random image subsets: 0.2 million, 2 million, 20 million, 200 million and 1.5 billion. The Precision@K curves were averaged over all 500 queries, which is shown in Fig.11(a). Fig.11(b) illustrates how *Precision@10* value changes with the growth of image sets — when the image data set grows up, the sketch-based retrieval performance improves significantly. This is reasonable, since more images provide a wider coverage of our visual world and can respond better to users' uncontrollable drawings.

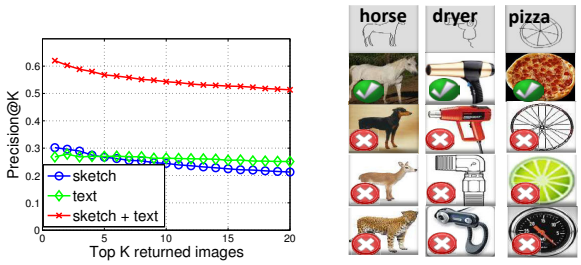
Enhancement on Keyword-based Image Retrieval

Since each query in *Sketch500* is associated with a keyword, which can be taken as a keyword query in a keyword-based search, we compared the retrieval performances on the 1.5 billion image data set under three schemes: **Text**, **Sketch** and **Sketch+Text**. As shown in Fig.12(a), **Sketch+ Text** significantly outperformed both **Text** and **Sketch**. For pure keyword-based search, since no shape information is considered, users are hard to find desired images with a particular shape. For pure sketch-based search, the "polysemy" of shapes will bring much noisy images with similar shapes but inconsistent concepts. E.g., the shape of a standing dog is very close to that of a standing horse (see Fig.12(b)). By combining both the shape and text, the search engine might satisfy those users who have more specific search intentions. The top results for some query sketches under these three schemes are compared in Fig.14.



(a) Precision@K (b) P@10 vs. Image Number

Figure 11: Retrieval performance vs. image set size. (a) Precision@K was evaluated for five randomly sampled data sets of 0.2M, 2M, 20M, 200M, and 1.5B images. (b) The plot shows how Precision@10 (in (a)) increases as the data set grows up.

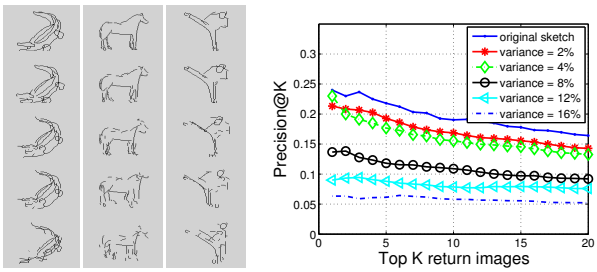


(a) Precision@K (b) Shape Polysemy

Figure 12: Enhancement on keyword-based image search. (a) Retrieval performance under Text, Sketch, Sketch+Text schemes. (b) Illustrative examples for "shape polysemy": since many objects might share similar shapes, image search purely based on the sketch often leads to ambiguous results. This can explain why combining both shape and keyword could significantly enhance retrieval accuracy.

Influence of Hand-drawn Sketch Quality

Since there might be large intra-class variations in human drawings, we'd like to see how the sketch quality impacts the system performance. We distorted the sketch queries by randomly and independently offsetting all strokes of a sketch to different directions. The moving distance is a parameter multiplied by the longer side of the sketch, and this parameter is called "variance" in this work. 100 sketches were randomly sampled from *Sketch500* in this experiment. Fig.13(b) shows the retrieval performances with distorted sketches under different distortion variances. We can see that, the retrieval performance consistently dropped as the increase of variance, which is not a surprise since sometimes even human vision are not easy to tell the distorted sketches very well, as shown in Fig.13(a).



(a) Sketch Qualities (b) Precision@K

Figure 13: Retrieval performance under different sketch qualities. (a) Three example columns indicating 5 different sketch quality levels with distorted variations ±2%, ±4%, ±8%, ±12%, ±16%. (b) Precision@K curves under different variations.

6. CONCLUSIONS

In this paper, we have presented our study on how to build a practical system to support sketch-based image search on a billion-level database, which might be the first attempt in both academia and industry. To achieve this goal, we systematically studied related problems such as feature selection, compact feature representation, fast matching and indexing, with particular considerations to some practical issues such as computation cost, memory cost, and retrieval speed. Currently this system is sensitive to affine translation. We will continue improving this system, based on which we hope other shape/sketch related applications which were thought very challenging might become feasible.

7. ACKNOWLEDGEMENTS

The work of Xinghai Sun and Chao Xu was partially supported by NBRPC 2011CB302400, NSFC 60975014, 61121002 and NSFB 4102024.

8. REFERENCES

[1] S. Belongie, J. Malik, and J. Puzicha. Shape context: A new descriptor for shape matching and object recognition. *NIPS*, 2001.

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

[3] Y. Cao, C. Wang, L. Zhang, and L. Zhang. Edgel index for large-scale sketch-based image search. In *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. In *ACM MM*, 2010.

[5] M. Eitz, J. Hays, and M. Alexa. How do humans sketch objects? *SIGGRAPH*, 2012.

[6] 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.

[7] M. Eitz, K. Hildebrand, T. Boubekeur, and M. Alexa. Sketch-based image retrieval: Benchmark and bag-of-features descriptors. *TVCG*, 2011.

[8] R. Hu, M. Barnard, and J. Collomosse. Gradient field descriptor for sketch based retrieval and localization. In *ICIP*, 2010.

[9] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *PAMI*, 2011.

[10] H. Jégou, R. Tavenard, M. Douze, and L. Amsaleg. Searching in one billion vectors: re-rank with source coding. In *ICASSP*, 2011.

Figure 14: Example queries and corresponding top results (from 1.5 billion images) for Tag+Sketch, Sketch, Tag. The green circle and blue triangle indicate the images that are not consistent with the query in terms of shape and concept respectively. The red cross indicate neither of them are consistent. It is clear that the results are greatly enhanced when combing the two search modalities.

[11] L. Paulevé, H. Jégou, and L. Amsaleg. Locality sensitive hashing: A comparison of hash function types and querying mechanisms. *Pattern Recognition Letters*, 2010.

[12] C. Siagian and L. Itti. Rapid biologically-inspired scene classification using features shared with visual attention. *PAMI*, 2007.

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

[14] Z. Sun, C. Wang, L. Zhang, and L. Zhang. Query-adaptive shape topic mining for hand-drawn sketch recognition. In *ACM MM*, 2012.

[15] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *PAMI*, 2008.

[16] K.-Y. Tseng, Y.-L. Lin, C.-Y. Hsiu, and W. H. Hsu. Sketch-based image retrieval on mobile devices using compact hash bits. In *ACM MM*, 2012.

[17] C. Wang, F. Jing, L. Zhang, and H.-J. Zhang. Scalable search-based image annotation of personal images. In *ACM MIR*, 2006.