MindFinder: Image Search by Interactive Sketching and Tagging

Changhu Wang
Microsoft Research Asia
No. 49, Zhichun Road
Beijing 100190, P.R.China
chw@microsoft.com

Zhiwei Li
Microsoft Research Asia
No. 49, Zhichun Road
Beijing 100190, P.R.China
zli@microsoft.com

Lei Zhang
Microsoft Research Asia
No. 49, Zhichun Road
Beijing 100190, P.R.China
leizhang@microsoft.com

ABSTRACT

In this technical demonstration, we showcase the MindFinder system – a novel image search engine. Different from existing interactive image search engines, most of which only provide image-level relevance feedback, MindFinder enables users to sketch and tag query images at object level. By considering the image database as a huge repository, MindFinder is able to help users present and refine their initial thoughts in their mind, and finally turn thoughts to a beautiful image(s). Multiple actions are enabled for users to flexibly design their queries in a bilateral interactive manner by leveraging the whole image database, including tagging, refining query by dragging and dropping objects from search results, as well as editing objects. After each action, the search results will be updated in real time to provide users up-to-date materials to further formulate the query. By the deliberate but easy design of the query, MindFinder not only tries to enable users to present on the query panel whatever they are imagining, but also returns to users the most similar images to the picture in users' mind. By scaling up the image database to 10 million, MindFinder has the potential to reveal whatever in users' mind, that is where the name *MindFinder* comes from.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Query formulation, Search process; H.5.2 [**User Interfaces**]: User-centered design

General Terms

Algorithms, Design, Experimentation

Keywords

MindFinder, interactive search, sketching, tagging

1. INTRODUCTION

With the prevalence of the Internet and digital cameras, effective and efficient image retrieval techniques have become an important research direction in both commercial and academic circles. There are mainly two basic problems in image retrieval. The first one is query formulation, that is how to interpret an *implicit query* in a user's mind such as "I want to find a scene in which a couple is standing together by the sea at sunset", as shown in Fig. 1, into an *explicit query* expressed by some features the computer can easily process. The second one is query matching, that is how to find the images that best fit for the explicit query. Since the solution to the second problem highly depends on the first problem, query formulation should be given primary importance in image retrieval.

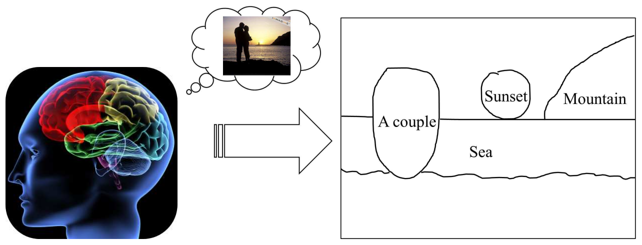


Figure 1: An implicit query in a user's mind: i.e. "I want to find a scene in which a couple are standing together by the sea at sunset."

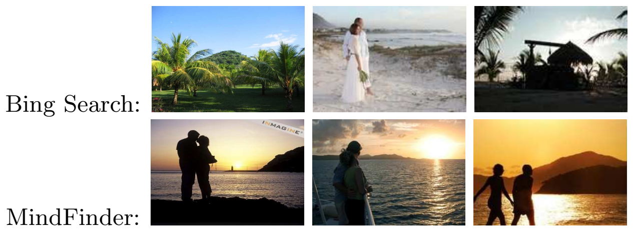


Figure 2: Top row: the top 3 search results of Bing Image Search using query "*sunset sea couple mountain*". Bottom row: the top 3 search results of MindFinder.

Currently, based on the types of query formulation methods, text-based and content-based image retrieval frameworks become very popular in commercial and academic circles. Although both these two kinds of frameworks have been widely studied and applied in commercial and academic systems, their query formulation methods are far from satisfactory for a user to express his/her boundless imagination. For example, a user may want to find a scene in which a couple is watching sunset by the sea, which is simply illuminated in Fig. 1. It is really not easy to search images similar to such a complex scene. The top results using keywords "*sunset sea couple mountain*" as the query in Bing Image Search[1] are shown in Fig. 2. It is quite clear that

[1]http://www.bing.com/images

the results are not satisfactory. For query by example, it is also difficult to interpret the user's mind by finding a query image for retrieval. While the ongoing work about specific image retrieval [1] seems far from practical applications.

In order to better formulate a user's implicit query in mind, some interactive techniques have been developed, most of which can be classified into two categories, i.e. *search result-based* interactive methods and *query-based* interactive methods. Search result-based interactive methods try to catch users' intentions by interactively refining the search results guided by users' interactions. Relevance feedback [2] is a typical approach in this category. Query-based interactive approaches become more and more popular in recent years, which try to enable more user interactions by providing certain attributes that could be specified by users. For example, Xcavator[2] enables users to draw points or lines on the query image, and then use them to emphasize key color features and their spatial relationships during search. Color-structured image search [3] enables users to draw a few color strokes to indicate the intent to improve search quality. SkyFinder [4] defines several attributes for sky image retrieval, which could be specified by users.

In spite of the success of existing interactive image search techniques, most of them are one-side interactive search and only consider how to leverage users' effort to catch their intentions, rather than help users to express their queries by leveraging the image database. Furthermore, most of them only use one type of interaction. For example, relevance feedback approaches only use interactive indication from users to tell whether those results are relevant or not. Xcavator and color-sturcture image search only involve visual content features. SkyFinder is particularly designed for sky image retrieval. Recently, Chen et al. [5] develop an image montage system, i.e. Sketch2Photo, to stitch several images together in agreement with the sketch and tags provided by users. In spite of the leverage of both sketching and tagging of the queries, users need to draw the implicit query in mind onto the query panel totally in one time and there is no user interaction at all. Moreover, the purpose of Sketch2Photo is to stitch images representing different objects into the resulting image, rather than to find images in the database to meet what in the user's mind.

In this work, we develop the MindFinder system, which is a bilateral interactive image search engine by interactive sketching and tagging. Different from existing interactive image search engines, most of which only provides *query-based* or *search result-based* interaction, MindFinder enables a *bilateral query$\leftrightarrow$search result* interactive search, by considering the image database as a huge repository to help users express their intentions. Moreover, MindFinder also enables users to tag during the interactive search, which makes it possible to bridge the semantic gap. Multiple actions are enabled for users to flexibly design their queries in a bilateral interactive manner by leveraging the whole image database, including tagging, refinding query by dragging and dropping objects from search results, as well as editing objects. After each action, the search results will be updated in real time to provide users up-to-date materials to further formulate query. Besides the contributions in the query formulation stage, in order to support the real time interactions between the system and users, a novel object-based indexing and retrieval algorithm is also developed for query matching. Therefore, MindFinder not only trys to enable users to present on the query panel whatever they imagine in their mind, but also returns to users the most similar images to the picture in users' mind. Fig. 2 shows the top 3 images retrieved by MindFinder according to the query in a user's mind shown in Fig. 1. In this technical demonstration, users can try their own searching on the MindFinder system.

[2]http://www.xcavator.net

2. SYSTEM OVERVIEW

In this section, we introduce the MindFinder system. Ten million images together with corresponding textual labels are indexed at back-end to support the interactions between users and the system, which will be introduced in Section 3.

There are two panels on the interface of MindFinder, i.e. *query panel* and *data panel*, out of which the query panel is used for users to design and edit the query and the data panel returns the interactive search results in real time for users' further exploitation. We still use the example query aforementioned to guide our tour on MindFinder system, that is, a user wants to find a scene in which a couple are standing together by the sea at sunset, which has been shown in Fig. 1.

We support the following actions to enable a user to interactively draw the scene in his/her mind onto the query panel: (See Fig. 3 for the contents of the two panels after each action. Notice that the data panel will be automatically updated in real time according to users' actions.)

1. **Tagging and Text-based Image Search**

   User: Draw a curve[3] at the top left corner of the query panel. Then type the tag "sunset" in it.

   System: Images related to sunset are displayed in the data panel in real time. In this stage, text-based image search technique is used.

2. **Object Drag-Drop and Object-based Image Search**

   User: Find an interesting image that contains a sun in the data panel. Then draw a curve around the sun, and drag this region and drop it to the region containing the tag "sunset" in the query panel. The size of the sun will automatically scale up or down to fit for the size of the region containing the tag "sunset".

   System: Images in the data panel are updated according to both the visual similarities between images in database and the sun in the query panel, as well as the distance between the spatial position of the sun in images and the sun in the query panel. Technical details in this part will be introduced in the next section.

3. **Object Spatial Transformation and Object-based Image Search**

   User: If the user is not satisfied with the position or the scale of the sun he just placed, he can translate, resize, or rotate the sun into another position or scale.

[3]The curve is unnecessary to be closed. Naive method could be used to close a curve.

Object removal is also enabled.

System: Similar to the system action of stage 2, the images are updated in real time.

4. **Object Drag-Drop, Spatial Transformation, and Object-based Image Search**

   User: If the user find there is one image containing both the sunset and the sea, he can directly crop some part of the sea and then drag-drop it into the query panel[4]. Then rotate, rescale, and place it to a proper position in the query panel.

   System: Similar to the system action of stage 2, the images are updated according to both the visual and spatial information of the two objects, i.e. the sun and the sea.

5. **Add Other Objects to the Query Panel and Find the Most Similar Images to the Query Panel**

   User: Similar to the stage 1 and 2, put objects "a couple" and "mountain" onto the query panel.

   System: Similar to the system action of stage 4, the images are updated according to the whole scene, objects, and tags in the query panel.

2.1 Discussion: Implicit Label Tool

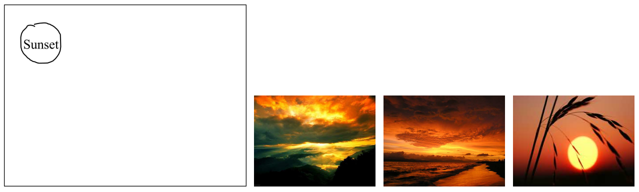
During the object drag-drop action after users add tags in the query panel, users have implicitly associate the object in the source image in data panel with the tags. Since in current standard image datasets, tags are usually associated with images rather than objects, and manually associating tags with objects are really tedious and time-consuming, our MindFinder system dose provide an effective method to naturally collect users' label information in object level when they search images using MindFinder.

3. TECHNICAL DETAILS

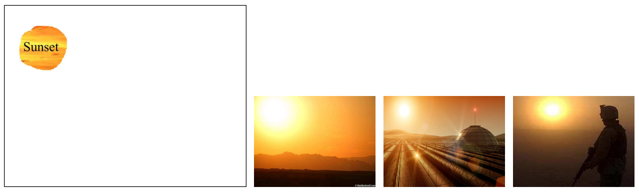
In order to support very specific queries, about ten million Panoramio[5] images together with corresponding textual tags are indexed by two 64-bit machines, each of which has 4 cores and 8GB memory. By a well designed index architecture, the system can respond user's query in 1 second. Common text retrieval techniques, such as filtering by inverted list and then reranking by cosine similarity of tf-idf features, are used to support text-based image search. These techniques are not discussed in this paper. For object-based image search, we propose an efficient and effective multi-model retrieval algorithm.

3.1 Local Feature-based Multi-Model Algorithm for Object-based Image Search

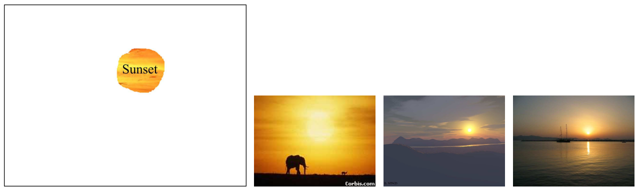
Because the query is expressed as multiple objects and tags with positional information in the query panel, and we want to find images with those objects at similar positions, we use a kind of local features to represent both the query and the images in the database. The detailed feature extraction and indexing process are shown in Fig. 4. The query panel is uniformly divided into 10×10 patches, and all the images in database are also divided into patches in the same manner. Each patch in an image has a $\{c_x, c_y\}$-coordinate to show its position in the image. For each patch, we extract three kinds of features: 44-dim correlogram, 6-dim color moment, and 14-dim texture moment features. On average, we got about 750 million patches for 10 million images. The amount of patches is extremely high. Most of existing indexing approaches cannot be applied to index



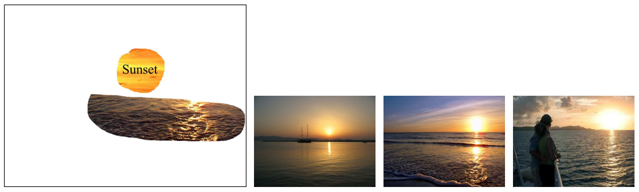
(1). Tagging and text-based image search.



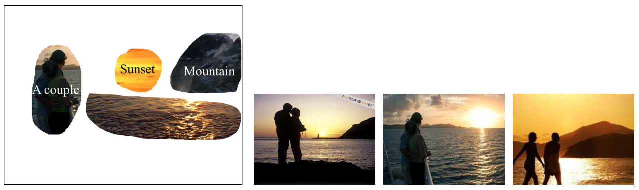
(2). Object drag-drop and object-based image search.



(3). Object spatial transformation and object-based image search.



(4). Object drag-drop, spatial transformation, and object-based image search.



(5). Add other objects to the query panel and find the most similar images to the query panel.

Figure 3: The query panel and the top three images in the data panel after each action.

[4]Of course, the user can repeat the user action of stage 1 and 2 to put object "sea" into the query panel. Here we achieve it in a different way.

[5]http://www.panoramio.com/

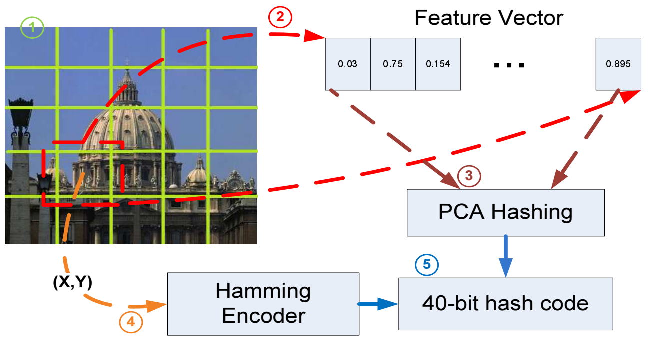


Figure 4: The proposed indexing approach consists of six steps: dividing images into local patches, extracting visual feature for patches, performing dimension reduction by PCA, encoding positions by a Hamming encoder, quantizing features to 0-1 bit vectors (i.e. hash code), and building inverted index.

them. Therefore we propose a novel indexing solution. The core of this solution is an algorithm to generate very compact and representative hash codes for patches, which is termed as *PCAHashing*. The *PCAHashing* algorithm consists of three steps:

1. Randomly sample 100K patches, and train a k-dim PCA model;
2. Apply the PCA matrix to reduce the dimensionality of all patch features in the database to be k-dim;
3. Quantize each component of a k-dim feature to be 1 if it is no less than 0, otherwise 0.

In this way, each patch is quantized as a k-bit 0-1 vector, which is termed as a hash code in this paper. Actually, this approach seeks to divide the feature space to be some fixed grids, i.e. quadrant. Patches in the same grid are encoded as the same hash code. By performing PCA, we find an orthogonal subspace for the original feature space. In the subspace, image patches symmetrically distribute around the axes which are orthogonal to each other. Thus by the quantizing rule, we are likely to get grids with almost equal number of patches in each of them. This property is critical for a hash algorithm and consequently indexing algorithms. In our system, k is set to be 32.

To incorporate the positional information of patches in the hash code, we concatenate 8 bits to the hash code of each patch, in which 4 bit are code for x coordinate, and the other 4 bits are for y coordinate. These bits are coded in a Hamming code manner. That is to say, if the Euclidean distance of two patches are bigger than 2, their Hamming distances will be bigger than 2. Therefore, we get a 40-bit hash code for each patch, and the Hamming distances of hash codes of patches can reflect both their visual similarities and spatial distances.

To make the on-line search process extremely efficient, we deem each hash code as a word as in text retrieval, and consequently build inverted index for them. To tolerate necessary variances within similar patches, for a hash code, h, (i.e. a word), we assign patches, whose Hamming distances to h are no bigger than 3, in h's inverted list. It is noted that, even if two patches are very similar in terms of visual content, they are far from each other spatially, they will be assigned into different lists. This property prevents patches at positions, which are far from the position of a query patch, from being returned as similar patches.

Let Q represent the query panel, and I represent an image in the database. We use $\mathcal{Q}_{\prime} = \{x_q\}$ to denote the set of patches that are partly or totally covered by the object regions in the query panel, and use w_q to denote the cover rate[6] of patch x_q. Let[7] $s(x_q, I) = \max_{x_i \in I} \exp^{-\|x_q - x_i\|_2}$. We can calculate the similarity between query panel Q and image I as follows:

$$sim(Q, I) = \sum_{x_q \in \mathcal{Q}_{\prime}} w_q * s(x_q, I) + \beta sim_{tag}(Q, I) \quad (1)$$

where $sim_{tag}(Q, I)$ is the similarity between the tags labeled in the query panel and the textual description of image I, in which cosine similarity is used. β is a trade-off parameter to balance the textual query and visual query[8]. We use $sim(Q, I)$ to rank images in our database and show the top-ranked images in the data panel.

4. CONCLUSIONS

The main contributions in this work can be summarized as follows:

1. We develop the MindFinder system, which is a bilateral interactive image search engine.
2. MindFinder enables multiple actions for users to flexibly design their queries in a bilateral interactive manner by leveraging the image collection in real time.
3. A novel local feature-based multi-model retrieval algorithm is proposed for object-based image search to ensure the system to return to users the most similar images to the picture in a user's mind.
4. MindFinder provides an effective method to naturally collect users' label information in object level when they search images using MindFinder.

5. REFERENCES

[1] C. Liu, D. Wang, X. Liu, C. Wang, L. Zhang, and B. Zhang. Robust Semantic Sketch based Specific Image Retrieval. *TechReport, Tsinghua University.*, http://166.111.138.19/paper/CailiangLIU_TR2010_SketchQuery.pdf, 2010.

[2] X. S. Zhou and T. S. Huang. Relevance feedback in image retrieval: A comprehensive review. *Multimedia systems*, 2003.

[3] J. Wang, X. S. Hua, and Y. Zhao. Color-structured image search. *TechReport, MSR-TR-2009-82*, 2009.

[4] L. Tao, L. Yuan, J. Sun. SkyFinder: attribute-based sky image search. *ACM SIGGRAPH*, 2009.

[5] T. Chen, M. M. Cheng, P. Tan, A Shamir, and S. M. Hu. Sketch2Photo: internet image montage. *ACM SIGGRAPH ASIA*, 2009.

[6]The cover rate is defined as the ratio of the area of the patch covered by the object region to that of the patch itself.
[7]In practice, to be efficient, we use x_q to retrieval the top k patches in the database. If all the patches of image I are not in the top k patches of x_q, $s(x_q, I)$ will be set to be zero.
[8]In practice, we set β to be a very large value to give the textual query higher priority.