Mobile Product Search with Bag of Hash Bits

Junfeng He, Tai-Hsu Lin, Jinyuan Feng, Shih-Fu Chang
Electrical Engineering Department, Columbia University
jh2700@columbia.edu, lord.lth@gmail.com, {jf2651, sfchang}@ee.columbia.edu

ABSTRACT

The advent of smart phones has provided an excellent platform for mobile visual search. Most of previous mobile visual search systems adopt the framework of "bag of words",in which words indicate quantized codes of visual features. In this work, we propose a novel mobile visual search system based on "bag of hash bits". Using new ideas for hash bit selection, multi-hash table generation, and hamming-distance soft scoring, we overcome the problem of bit inefficiency affecting the traditional hashing approaches, and achieve promising accuracy outperforming state of the art. The framework is also general in that general feature type can be used for generating the hash bits. Demos and experiments over a large scale product image set demonstrate the effectiveness of our approach.

Categories and Subject Descriptors

H.3 [**INFORMATION STORAGE AND RETRIEVAL**]: Information Search and Retrieval

General Terms

Algorithms

Keywords

Mobile Visual Search, Image Search and Retrieval

1. INTRODUCTION

The advent of smartphones provides a perfect platform for mobile visual search, in which many interesting applications have been developed, such as mobile product search (Google Goggles, Amazon Snaptell), location search or augmented reality,etc. For mobile visual search, local feature(LF) like SIFT or SURF etc., is a popular choice, since global features usually can not support object-level partial search or match, which is crucial in applications like product search.

One unique challenge of mobile visual search is to reduce the amount of data sent from the mobile to the server because of limited bandwidth of networks such as 3G. Many mobile visual systems extract features in the mobile side. However, such local features need to be compressed before transmission; Since sending the raw local features may cost more than sending the image. Typical ways for compressing local features is to quantize them to visual words, or apply novel coding like CHoG[1]. On the server side, most systems adopt the model of "bag of words" (BoW), which represents one image as histograms of visual words contained in the image. Usually a subset of candidate images would be chosen from the database , according to some kinds of distance (e.g., L_2) between the query BoW histogram and the histgorams of database images. Then some geometric reranking/verification such as RANSAC, will be applied to rerank the candidates.

MM'11, November 28–December 1, 2011, Scottsdale, Arizona, USA.
ACM 978-1-4503-0616-4/11/11.

2. PROPOSED SYSTEM

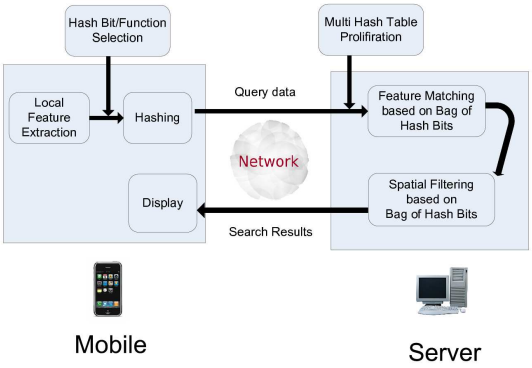

Figure 1: Architecture of the Proposed Bag of Bits Mobile Search System.

We have developed new algorithms and a visual search system based on bag of hash bits instead of conventional bag of words. Figure 1 shows the overall workflow of the system. First, on the client (mobile) side, we compress local features to tens of bits by hashing, which meet the bandwidth requirements in practical mobile networks. On the server side, after receiving hash bits of each local feature, we can represent the whole image as bag of hash bits. To some extend, the hash bits of each local feature can also be viewed as visual word index, however, the advantage of using hash bits are: the hamming distance between "word index" of hash bits, can be used to approximate the feature distance, and moreover, the hash bits allows us to create indexing structure in a very flexible manner. These advantages are important in developing successful search systems that minimize the transmission delay and maintain high retrieval accuracy. With matched local features, we then apply an image-level spatial filtering to re-rank the images and choose the top returned results. The spatial filtering utilizes both the hash bits and the location of local features in one image.

Some previous works have reported that hashing may not

be a good choice for mobile visual search, since a large number of bits are needed for each local feature, hence causing high much transmission cost. In this work, however, we introduce several novel ideas (summarized below with details in the next section).

1. Reducing the number of required hash bits on the mobile side by balancing and selecting hash bits[3].
2. Generating multiple hash tables on the server side by re-using a compact pool of hash bits.
3. Leveraging the hamming distance provided by hash bits in the spatial verification/filtering step.

Our system will have similar bit budgets (and hence transmission cost) as the state-of-art works like CHoG [1], but a higher search accuracy.

3. DESCRIPTIONS

3.1 Mobile side–Hash Local Feature into Bits

In the current implementation, we choose SURF as the local feature, of its proven performance in accuracy and speed in several existing systems.. We apply the Locality Sensitive Hahsing(LSH)method [2] to compress the SURF features. Following the considerations in [1], constrained by transmission speed, we limit to 80 hash bits for each local feature, so the hash bits must be very compact. We balance each hash bit so that half of bits produced by each hashing function are +1, and the other -1. To further improve the efficiency and compactness of the hash bits, we applied a bit selection algorithm as described [3], which chooses 80 most compact bits from an initial candidate set of 1024 LSH bits. As shown in [3, 4], bit balancing and selection can greatly reduce the number of hash bits needed. More details about the hash bit selection algorithm can be found in [3].

3.2 Server Side–Search with Bag of Bits

3.2.1 *Matching Local Features with Hash Bits*

For one local feature in the query image, we need to find out its nearest neighbors from all local features stored in the database. One popular technique in improving the hashing performance (especially the recall rate) is to use multiple hash tables. We adopt the same idea, but instead of sending multiple sets of hash bits over the mobile network, we randomly select multiple subsets of bits from a single pool of hash bits (80 bits in the current implementation) and use each subset to construct a table. We can thus construct multiple tables without increasing the total amount of bits needed to be transmitted. We search each table by checking the buckets within a hamming radius to the query, instead of checking only the same bucket of the query, which can further reduce the number of tables needed. Matched images retrieved from each table are combined to form the candidate set for subsequent verification. In our system, we choose to use 12 tables, each indexed with 36 bits randomly drawn from the 80 bits pool aforementioned, with hamming radius $r = 1$ to 3, according to different tradeoffs between accuracy and time.

3.2.2 *Spatial Filtering*

Given the set of features in the database that were matched to features in the query image, one would like to check if these matches are geometrically consistent, i.e., whether a valid geometric transformation can be established between the feature positions on the query image and the positions in the database image. Considering the popular geometric verification method, RANSAC, is too slow, recently, a method based on length ratios [1] has been proposed to efficiently verify geometric correspondences without actually computing the transformation. Intuitively, it estimates the portion of matched features between query and reference images that share a consistent scale change. The higher value this is, the higher score the candidate reference image receives. Leveraging the hamming distance enabled by the hash bits, we further modify this step by placing higher weights on matched feature pairs with smaller hamming distances when computing the dominant scale change between query and reference images. This is in contrast to the previous method that only counts the frequency of match pairs. Our experiments confirmed clear performance improvement by this modification step incorporating the hamming distance of the hash bits.

4. PROTOTYPE AND EXPERIMENTS

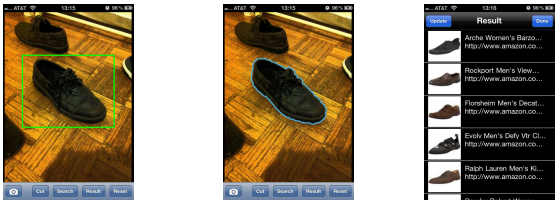

Figure 2: Example user interfaces. Users may select the while image, or a subwindow as query input, which can further be refined by interactive object segmentation tools like Grabcut. Matched products are shown on the right.

We have collected a product set with around 360K images crawled from Amazon.com. It contains diverse categories such as DVD/book covers, shoes, furnitures, groceries, kitchen supplies, etc. Each image contains about 100 SURF features on average, and hence we have about 36M local features in the database.

We have created a test set of more than 100 query images, which are separated from the reference database. The speed of our system is comparable to the state-of-the-art mobile visual search system, such as the one based on CHoG [1]. More specifically, the speed for the SURF feature extraction (1-1.5s), transmission of query hash bits(1-2s), search (1-3sec), result download/display (1-2s). However, our method can perform significantly better, with at least 20% improvement on precision of the top 100 search results, than the CHoG method [1] (i.e., compressing local features with CHoG in mobile side, and then applying BoW with a vocabulary tree of 1M codewords, and reranking with the spatial filtering in the server side).

5. REFERENCES

[1] V. Chandrasekhar, et.al. Mobile Product Recognition. *In Proceedings of ACM MM*, 2010.

[2] M. Charikar. Similarity estimation techniques from rounding algorithms. *In Proceedings of STOC*, 2002.

[3] J. He, T.-H. Lin, and S.-F. Chang. Hash Bit Selection for Large Scale Similarity Search. *Technical Report, Columbia University*, Sept., 2011.

[4] J. He, R. Radhakrishnan, S.-F. Chang, C. Bauer. Compact Hashing with Joint Optimization of Search Accuracy and Time. In Proceedings of CVPR, 2011.