Mobile Product Recognition

Sam S. Tsai, David Chen, Vijay Chandrasekhar
Gabriel Takacs, Ngai-Man Cheung, Bernd Girod
Department of Electrical Engineering, Stanford University
Stanford, CA 94305, U.S.A.
{sstsai, dmchen, vijayc, gtakacs, ncheung, bgirod}@stanford.edu

Ramakrishna Vedantham
Radek Grzeszczuk
Nokia Research Center
Palo Alto, CA 94305, U.S.A.
{ramakrishna.vedantham,radek.grzeszczuk}@nokia.com

ABSTRACT

We present a mobile product recognition system for the camera-phone. By snapping a picture of a product with a camera-phone, the user can retrieve online information of the product. The product is recognized by an image-based retrieval system located on a remote server. Our database currently comprises more than one million entries, primarily products packaged in rigid boxes with printed labels, such as CDs, DVDs, and books. We extract low bit-rate descriptors from the query image and compress the location of the descriptors using location histogram coding on the camera-phone. We transmit the compressed query features, instead of a query image, to reduce the transmission delay. We use inverted index compression and fast geometric re-ranking on our database to provide a low delay image recognition response for large scale databases. Experimental timing results on different parts of the mobile product recognition system is reported in this work.

Keywords

mobile visual search, content-based image retrieval

1. INTRODUCTION

We present a fast and scalable mobile product recognition system for the camera-phone. Consider a customer seeing a book in a bookstore and wanting to learn more about the book. Using our system, the customer takes a snapshot of the book with the camera-phone. The camera-phone processes the query image and sends query data over the wireless network to a remote server. The server processes the query data and recognizes the item among a large database of products. Then, reviews of the book, pricing from other merchants, and electronic version availability of the book can be returned to the customer. For a customer in a music shop, a picture of a CD cover can be used as a query to find music samples for the particular product. If the query item is a DVD, a short description of the movie and its online availability is provided.

MM'10, October 25–29, 2010, Firenze, Italy.

The camera-phone is a convenient visual input device, having the ability to take photos and process data. These abilities enable many interesting image retrieval applications. For small image databases, we can store the database on the phone and run local image retrieval applications. For large image databases, the database is placed on a server with enough storage space. Query data is sent from the camera-phone to the server where image retrieval is performed and meta-data is prepared. Current mobile image retrieval systems on the market include SnapTell[3], Kooaba[2], Google Goggles[1]. However, they all suffer from a long lag between snapping a picture and the response to the query, typically 10s of seconds. Our system is much faster.

Communication between the camera-phone and the remote server requires a wireless network connection. The connection is typically EDGE, 3G, or WiFi, with 3G being the most prevalent. To send data over the wireless network, one wishes to send as little data as possible. We extract low bit-rate local descriptors (CHoG) from the query image [6, 5] and compress their locations using location histogram coding [13]. We send the significantly smaller compressed features instead of the query image, reducing the time spent in communication.

A large-scale database on the server suffers from large memory usage. Memory swapping operations slow down the recognition. We compress the inverted index in RAM to reduce the memory size and prevent memory swapping operations [8]. Complex geometric verification processes also increase the response time. We use fast geometric re-ranking to reduce the time for geometric verification [14]. Our system achieves image recognition less than one second and can provide an interactive experience to the user.

Compared to our previous system [12], we have reduced query data size by using better compression techniques, optimized image processing implementation on the phone, and improved our recognition engine to accommodate a database of beyond one million items. In this paper, we briefly review our mobile product recognition system in Sec. 2. We report the processing time of our system and evaluate the benefits of our approach in Sec. 3. In addition to this paper, a video presentation is provided[1].

2. SYSTEM OVERVIEW

Our mobile production recognition system is a local feature based visual search system. For this type of systems [3, 2, 12], the query image is converted into a sparse set

[1]http://msw3.stanford.edu/∼sstsai/mpr-demo.wmv

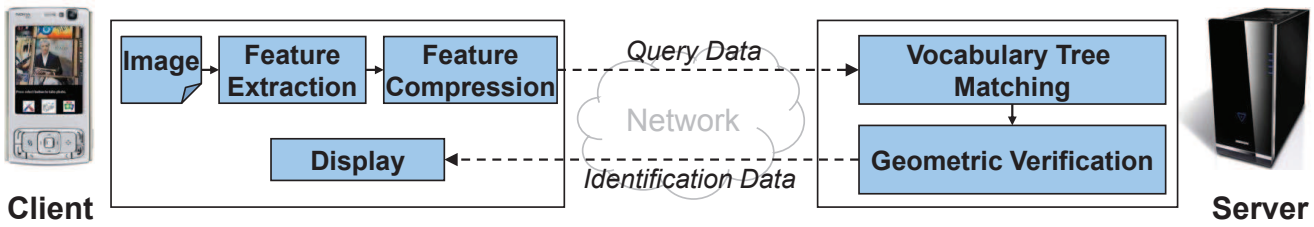

Figure 1: A client-server based mobile visual search system. Due to the large database, the image recognition server is placed at a remote location. In most systems [2, 3, 1], the query image is sent to the server and feature extraction is performed. In our system, we show that by performing feature extraction on the phone, we can significantly reduce the transmission delay and provide an interactive experience.

of local features and database images are retrieved using these features. From the image, interest points are found at locations with robust image structures [9, 10]. Then, a descriptor, such as SIFT [9], SURF [4], or CHoG [5, 7], is generated from a local image patch around each interest point. When retrieving from the database, the set of descriptors is quantized using a vocabulary tree (VT) [11], generating a histogram of visual words (HoVW). The query HoVW is compared to a set of database HoVW determined by the inverted index [11]. Finally, geometric verification (GV) is applied to the most similar matches. GV finds a coherent spatial pattern between features of the query image and the features of the candidate image. This ensures that the match is correct. Due to the large number of entries, it is not practical to store the database at the client. Thus, for a mobile visual search system, the client-server system architecture as shown in Fig. 1 is adopted.

Our system enables interactive user experience with very short response time. On the client side, we have optimized the feature extraction algorithm. We communicate with the server using low bit-rate descriptors (CHoG [5]) and compressed location information [13]. On the server side, we use inverted index compression [8] to reduce memory swapping operations which are typical in large-scale databases. We further use fast geometric re-ranking to reduce total GV processing time [14]. We describe these contributions of our system in the following sections.

2.1 Feature Extraction

The camera-phone has less compute power, typically only a tenth of what a desktop PC provides. In our previous system, feature extraction on the camera-phone took more than a few seconds [12]. To reduce the response time, we optimize the feature extraction implementation. We convert our feature extraction algorithm to use fixed-point arithmetics. Then, we store precomputed weighting values for soft-binning CHoG descriptor, reducing the amount of numerical computations. We successfully reduce the feature extraction time to approximately one second for a Nokia 5800 phone with 300MHz CPU.

2.2 Feature Compression

We send compressed query features, which is typically an order magnitude smaller than the query image, from the camera-phone to the server. On the phone, we extract CHoG descriptors from the query image. CHoG is a low bit-rate descriptor designed bottom up with compression in mind. It provides up to 20× bit-rate saving over SIFT and has comparable image retrieval performance to SIFT [7]. We compress the location of the descriptors in the query image using location histogram coding [13]. With this technique, location information requires only about 6 additional bits per descriptor; most of the information is conveyed by the ordering of the feature descriptors. Overall, the size of the compressed query features is $2 \sim 4$ kilobytes.

2.3 Inverted Index Compression

Inverted index compression enables large-scale image retrieval on the server. For each leaf node in the VT, the inverted index stores a list of image IDs indicating which database images have features that were quantized to the node, as well as the number of features quantized to the node. When retrieving an image from the database, the query features are quantized to the leaf nodes. Database images in the list of these leaf nodes are marked as candidate matches to the query image. Histogram of visual word comparison is performed on this smaller set of candidates. For an one million image database, the inverted index can take up several giga-bytes of RAM [8]. By compressing the inverted index, we are able to achieve up to six-fold reduction in memory usage. The memory usage reduction decreases the number of memory swap operations and reduces the server response time.

2.4 Fast Geometric Re-ranking

We incorporate fast geometric re-ranking in our system to shorten the list of candidates for the complex geometric verification [14]. A geometric similarity score of the query image and the candidate image is generated. This score can be computed efficiently by comparing the geometric properties of the VT visual word matches. We calculated the score for a few hundred images and keep only the top ten images for GV. We reduce the set of images that requires comprehensive GV and achieve a response time that is less than 1 second.

3. SYSTEM IMPLEMENTATION

We build a product recognition database of more than one million entries from an online product database. We deploy our recognition server on a Linux server with a Xeon E5410 2.33GHz CPU and 32GB of RAM. For the client side, we build a graphical user application for S60 Nokia phones and include features such as photo querying, product sampling, product purchasing, and query history browsing (see Fig. 2). For comparison, we include two different modes of operation. In *Send Feature* mode, we process the query image on the phone and transmit compressed query features to the server. In *Send Image* mode, we transmit the query image

(a) (b) (c)

Figure 2: Graphical user interface in developed in Qt. (a) shows the photo capturing screen where the user can take a snapshot of a product. (b) shows the product information that the server returns to the user after the product is recognized. If connection to a store is available, the shopping can be processed on the phone. (c) shows the list of previous queries that the user has previously taken, providing a history of products that the user has been interested in.

to the server and all operations are performed on the server. We present the timings of the two modes in the following sections. The processing time on the client is measured using a Nokia 5800 phone with a 300MHz CPU.

3.1 Processing Time

We show the time of different operations performed on the client and on the server in Table 1. While 0.9-1.4 seconds more time is required for *Send Feature* mode to extract features on the phone, sending compressed query features requires much lesser transmission time than sending the query image. Using VT matching with compressed inverted index, we can search through a million image database in 100 milliseconds. We perform GV on a short list of 10 candidates after fast geometric re-ranking of 250 candidates.

Table 1: Processing Time

Client side operations	Time (sec)
Image capture	1-2
Feature extraction and compression (for *Send Feature* mode)	1-1.5
Server side operations	Time (msec)
Feature extraction (for *Send Image* mode)	100
Vocabulary tree matching	100
Fast geometric re-ranking (per image)	0.46
Geometric verification (per image)	30

3.2 Transmission Time

We present experimental results of sending data over the 3G wireless network. We use query data size ranging from typical compressed query features (2 KB) to a typical query image (50 KB) to learn how different sized queries affects transmission time. We establish a communication link between different transmissions and set the communication time-out time to 60 seconds. We run the experiment continuously over several days and collect results of more than 5000 transmissions requests. We test at three different locations, each resembling a place where a user might access network.

The median and average transmission latency of our experiments are shown in Fig. 3. We see that sending the compressed query features typically takes approximately 3 seconds. The time required to send the compressed query image is up to 10 seconds, and varies significantly at different locations. Since any transmission that is above 60 seconds is discarded due to time-out, we show the percentage of transmissions that experienced a time-out in Fig. 4. We see that the success rate of sending compressed query features is better than the sending the compressed query image.

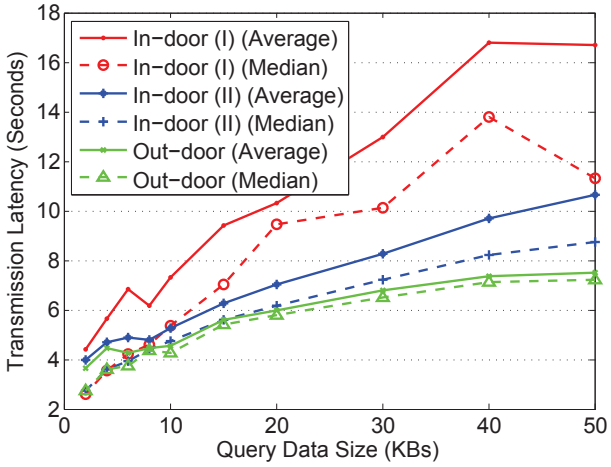

Figure 3: The transmission latency of different sized queries for three different locations. "In-door (I)" is tested in-doors with poor connectivity. "In-door (II)" is tested in-doors with good reception. "Out-door" is tested outside of buildings.

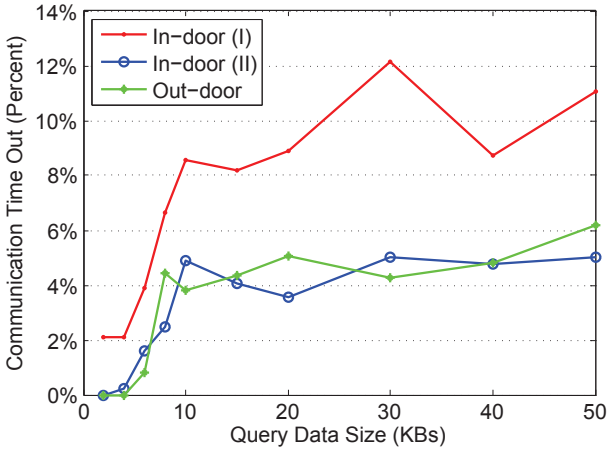

Figure 4: The communication time out percentage of different sized queries for three different locations. "In-door (I)" is tested in-doors with poor connectivity. "In-door (II)" is tested in-doors with good reception. "Out-door" is tested outside of buildings.

3.3 Timing Comparison

We compare the response time of the *Send Image* mode and the *Send Feature* mode of our current system. We also compare these results to the response time of the *Send Image* from our previous system. In *Send Feature* mode, feature extraction takes place on the phone and requires typically ∼1.2 second. In *Send Image* mode, no processing is needed on the phone. We account for the network transmission latency using experimental results from Sec. 3.2. We use the median transmission latency of "In-door (II)", which represents locations such as stores or shopping centers. In *Send Image* mode, we assume we need to transmit a query image of 40 KB. *Send Feature* mode is assumed to transmit compressed query features of 2 KB. On the server, we assume 1 second of total server processing time to additionally include I/O operations and meta-data preparation.

The response time of the *Send Image* mode in our previous system [12] is shown in Fig. 5 (a), representing a basic image retrieval system. Large memory requirements requires additional memory swapping operations, resulting in longer image recognition response time. We use inverted index compression to alleviate memory congestion and reduce the image recognition response time. The improvements are reflected in the response time of our current system, shown in Fig. 5 (b). However, due to the network transmission latency, the total response time is still around 10 seconds. The network transmission latency is decreased by reducing the size of the query. In Fig.5 (c), we show the response time of the *Send Features* mode, where we extract and send compressed query features. For *Send Features* mode, the network latency is ∼3 seconds and the overall response time is ∼5 seconds. Thus, using *Send Features* mode, we can reduce up to 5 seconds of network latency on 3G wireless network, providing the user with a much more interactive experience.

4. SUMMARY

We have developed a fast and scalable mobile product recognition system for the camera-phone. Our system is based on a low bit-rate image retrieval system with a client-server architecture. On the camera-phone, we extract features from the query image the user has taken in a couple of seconds. We extract low bit-rate CHoG descriptors, and compress the location information of the descriptors using local histogram coding. We send compressed query features over the wireless network. Sending the compressed features saves up to 5 seconds when compared to sending a query image. On the server, we perform vocabulary tree matching followed by geometric verification. We use inverted index compression to lessen the memory loading and fast geometric re-ranking to speed up the image recognition response. We show that by using our approach, we can reduce the total response time of mobile product recognition and provide an interactive experience to the user.

5. REFERENCES

[1] Google goggles. http://www.google.com/mobile/goggles/.

[2] Kooaba. http://www.kooaba.com.

[3] Snaptell. http://www.snaptell.com.

[4] H. Bay, T. Tuytelaars, and L. V. Gool. SURF: speeded up robust features. In *European Conference on Computer Vision*, pages 404–417, Graz, Austria, May 2006.

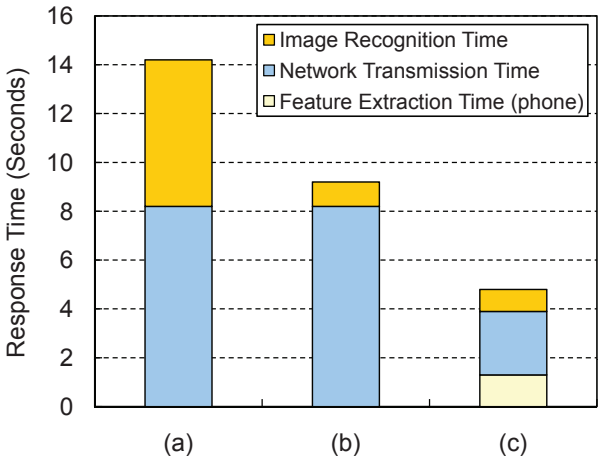

Figure 5: System response time of different modes of the basic image-retrieval system [12] and our current system. (a) shows the timing for the *Send Image* mode in [12]. It takes longer time to respond due to memory congestion. (b) shows the *Send Image* mode of our current system. The image recognition delay is reduced by incorporating inverted index compression. (c) shows the *Send Feature* mode. By compressing the features, we reduce the transmission delay and can provide an interactive experience.

[5] V. Chandrasekhar, G. Takacs, D. Chen, S. S. Tsai, R. Grzeszczuk, and B. Girod. CHoG: Compressed Histogram of Gradients. In *Conference on Computer Vision and Pattern Recognition*, 2009.

[6] V. Chandrasekhar, G. Takacs, D. Chen, S. S. Tsai, J. Singh, and B. Girod. Transform coding of feature descriptors. In *SPIE Visual Communications and Image Processing*, San Jose, CA, USA, January 2009.

[7] V. Chandrasekhar, G. Takacs, D. M. Chen, S. S. Tsai, Y. Reznik, R. Grzeszczuk, and B. Girod. Quantization schemes for CHoG. In *International Workshop on Mobile Vision*, 2010.

[8] D. M. Chen, S. S. Tsai, V. Chandrasekhar, G. Takacs, R. Vedantham, R. Grzeszczuk, and B. Girod. Inverted index compression for scalable image matching. In *Data Compression Conference*, Snowbird, UT, USA, 2010.

[9] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[10] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, October 2004.

[11] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Conference on Computer Vision and Pattern Recognition*, pages 2161–2168, New York, NY, USA, June 2006.

[12] S. S. Tsai, D. Chen, J. Singh, and B. Girod. Rate-efficient, real-time CD cover recognition on a camera-phone. In *ACM International Conference on Multimedia*, Vancouver, Canada, October 2008.

[13] S. S. Tsai, D. M. Chen, G. Takacs, V. Chandrasekhar, R. Vedantham, R. Grzeszczuk, and B. Girod. Location coding for mobile image retrieval. In *Proc. 5th International Mobile Multimedia Communications Conference*, 2009.

[14] S. S. Tsai, D. M. Chen, G. Takacs, V. Chandrasekhar, R. Vedantham, R. Grzeszczuk, and B. Girod. Fast geometric re-ranking for image-based retrieval. In *International Conference on Image Processing*, 2010.