Bernd Girod, Vijay Chandrasekhar, David M. Chen, Ngai-Man Cheung, Radek Grzeszczuk, Yuriy Reznik, Gabriel Takacs, Sam S. Tsai, and Ramakrishna Vedantham

Mobile Visual Search

[Linking the virtual and physical worlds]

Mobility in Media Search

Mobile phones have evolved into powerful image and video processing devices equipped with high-resolution cameras, color displays, and hardware-accelerated graphics. They are also increasingly equipped with a global positioning system and connected to broadband wireless networks. All this enables a new class of applications that use the camera phone to initiate search queries about objects in visual proximity to the user (Figure 1). Such applications can be used, e.g., for identifying products, comparison shopping, finding information about movies, compact disks (CDs), real estate, print media, or artworks. First deployments of such systems include Google Goggles [1], Nokia Point and Find [2], Kooaba [3], Ricoh iCandy [4]–[6], and Amazon Snaptell [7].

Mobile image-retrieval applications pose a unique set of challenges. What part of the processing should be performed on the mobile client, and what part is better carried out at the server? On the one hand, transmitting a Joint Photographic Experts Group (JPEG) image could take few seconds over a slow wireless link. On the other hand, extraction of salient image features is now possible on mobile devices in seconds. There are several possible client–server architectures.

- The mobile client transmits a query image to the server. The image-retrieval algorithms run entirely on the server, including an analysis of the query image.

Digital Object Identifier 10.1109/MSP.2011.940881
Date of publication: 15 June 2011

1053-5888/11/$26.00©2011IEEE

- The mobile client processes the query image, extracts features, and transmits feature data. The image-retrieval algorithms run on the server using the feature data as query.
- The mobile client downloads data from the server, and all image matching is performed on the device.

One could also imagine a hybrid of the approaches mentioned above. When the database is small, it can be stored on the phone, and image-retrieval algorithms can be run locally [8]. When the database is large, it has to be placed on a remote server and the retrieval algorithms are run remotely.

In each case, the retrieval framework has to work within stringent memory, computation, power, and bandwidth constraints of the mobile device. The size of the data transmitted over the network needs to be as small as possible to reduce network latency and improve user experience. The server latency has to be low as we scale to large databases. This article reviews the recent advances in content-based image retrieval with a focus on mobile applications. We first review large-scale image retrieval, highlighting recent progress in mobile visual search. As an example, we then present the Stanford Product Search system, a low-latency interactive visual search system. Several sidebars in this article invite the interested reader to dig deeper into the underlying algorithms.

[MOBILE IMAGE-RETRIEVAL APPLICATIONS POSE A UNIQUE SET OF CHALLENGES.]

ROBUST MOBILE IMAGE RECOGNITION

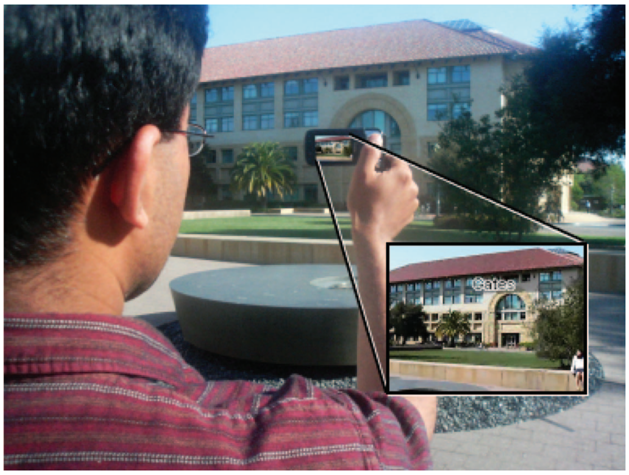
Today, the most successful algorithms for content-based image retrieval use an approach that is referred to as bag of features (BoFs) or bag of words (BoWs). The BoW idea is borrowed from text retrieval. To find a particular text document, such as a Web page, it is sufficient to use a few well-chosen words. In the database, the document itself can be likewise represented by a bag of salient words, regardless of where these words appear in the text. For images, robust local features take the analogous role of visual words. Like text retrieval, BoF image retrieval does not consider where in the image the features occur, at least in the initial stages of the retrieval pipeline. However, the variability of features extracted from different images of the same object makes the problem much more challenging.

A typical pipeline for image retrieval is shown in Figure 2. First, the local features are extracted from the query image. The set of image features is used to assess the similarity between query and database images. For mobile applications, individual features must be robust against geometric and photometric distortions encountered when the user takes the query photo from a different viewpoint and with different lighting compared to the corresponding database image.

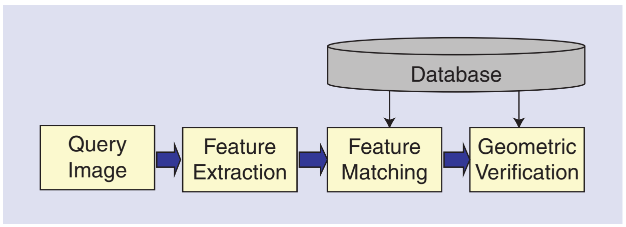
Next, the query features are quantized [9]–[12]. The partitioning into quantization cells is precomputed for the database, and each quantization cell is associated with a list of database images in which the quantized feature vector appears somewhere. This inverted file circumvents a pairwise comparison of each query feature vector with all the feature vectors in the database and is the key to very fast retrieval. Based on the number of features they have in common with the query image, a short list of potentially similar images is selected from the database.

Finally, a geometric verification (GV) step is applied to the most similar matches in the database. The GV finds a coherent spatial pattern between features of the query image and the candidate database image to ensure that the match is plausible. Example retrieval systems are presented in [9]–[14].

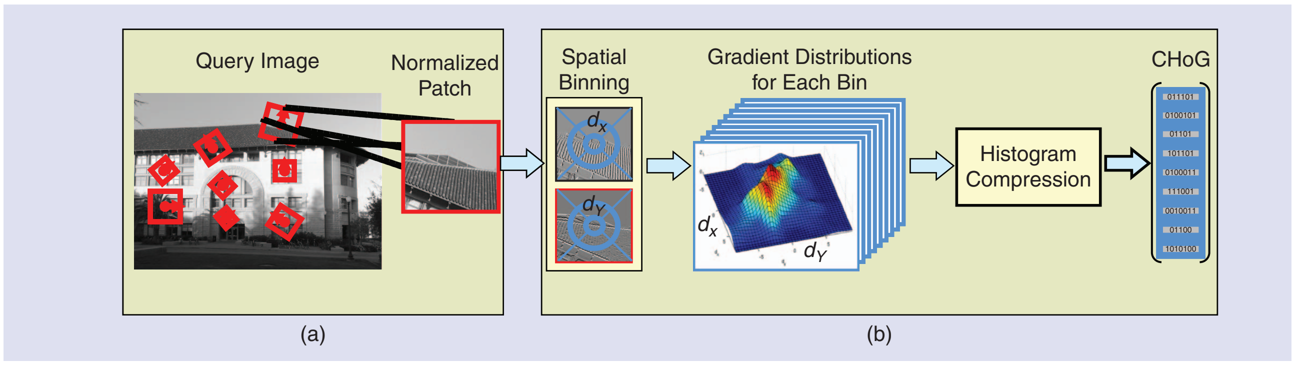
For mobile visual search, there are considerable challenges to provide the users with an interactive experience. Current deployed systems typically transmit an image from the client to the server, which might require tens of seconds. As we scale to large databases, the inverted file index becomes very large, with memory swapping operations slowing down the feature-matching stage. Further, the GV step is computationally expensive and thus increases the response time. We discuss each block of the retrieval pipeline in the following, focusing on how to meet the challenges of mobile visual search.



[FIG1] A snapshot of an outdoor mobile visual search system being used. The system augments the viewfinder with information about the objects it recognizes in the image taken with a camera phone.



[FIG2] A Pipeline for image retrieval. Local features are extracted from the query image. Feature matching finds a small set of images in the database that have many features in common with the query image. The GV step rejects all matches with feature locations that cannot be plausibly explained by a change in viewing position.



[FIG3] Illustration of feature extraction. We first compute interest points (e.g., corners and blobs) at different scales. The patches at different scales are oriented along the dominant gradient. Feature extraction is followed by computation of feature descriptors that capture the salient characteristics of the image around the interest point. Here, we illustrate how the CHoG descriptor is computed. The scaled and oriented canonical patches are divided into localized spatial bins, which gives robustness to interest-point localization error. The distribution of gradients in each spatial bin is compressed to obtain a very compact description of the patch. (a) Interest-point detection. (b) Computation of feature descriptors.

FEATURE EXTRACTION

INTEREST-POINT DETECTION

Feature extraction typically starts by finding the salient interest points in the image. For robust image matching, we desire interest points to be repeatable under perspective transformations (or, at least, scale changes, rotation, and translation) and real-world lighting variations. An example of feature extraction is illustrated in Figure 3. To achieve scale invariance, interest points are typically computed at multiple scales using an image pyramid [15]. To achieve rotation invariance, the patch around each interest point is canonically oriented in the direction of the dominant gradient. Illumination changes are compensated by normalizing the mean and standard deviation of the pixels of the gray values within each patch [16].

Numerous interest-point detectors have been proposed in the literature. Harris Corners [17], scale-invariant feature transform (SIFT) difference-of-Gaussian (DoG) [15] key points, maximally stable extremal regions (MSERs) [18], Hessian affine [16], features from accelerated segment test **(FAST)** [19], and Hessian blobs [20] are some examples. The different interest-point detectors provide different tradeoffs in repeatability and complexity. SIFT DoG and other affine interest-point detectors are slow to compute but are highly repeatable. The speeded up robust feature (SURF) interest-point detector provides significant speed up over DoG interest-point detectors by using box filters and integral images for fast computation. However, the box filter approximation causes significant anisotropy, i.e., the matching performance varies with the relative orientation of query and database images [21]. The FAST corner detector is an extremely fast interest-point detector that offers very low repeatability. In [22], Mikolajczyk et al. compare the different interest-point detectors in a common framework.

The Stanford Product Search system can perform feature extraction and compression on the client to reduce system latency. Current generation smartphones have limited compute power, typically only a tenth of what a desktop personal computer provides. We require interest points that are fast to compute and highly repeatable. We choose the Hessian-blob detector sped up with integral images [20], which provide a good tradeoff of repeatability and complexity. For video graphics array (VGA) images, Hessian-blob interest-point detection can be carried out in approximately 1 s on current-generation smartphones [14].

FEATURE DESCRIPTOR COMPUTATION

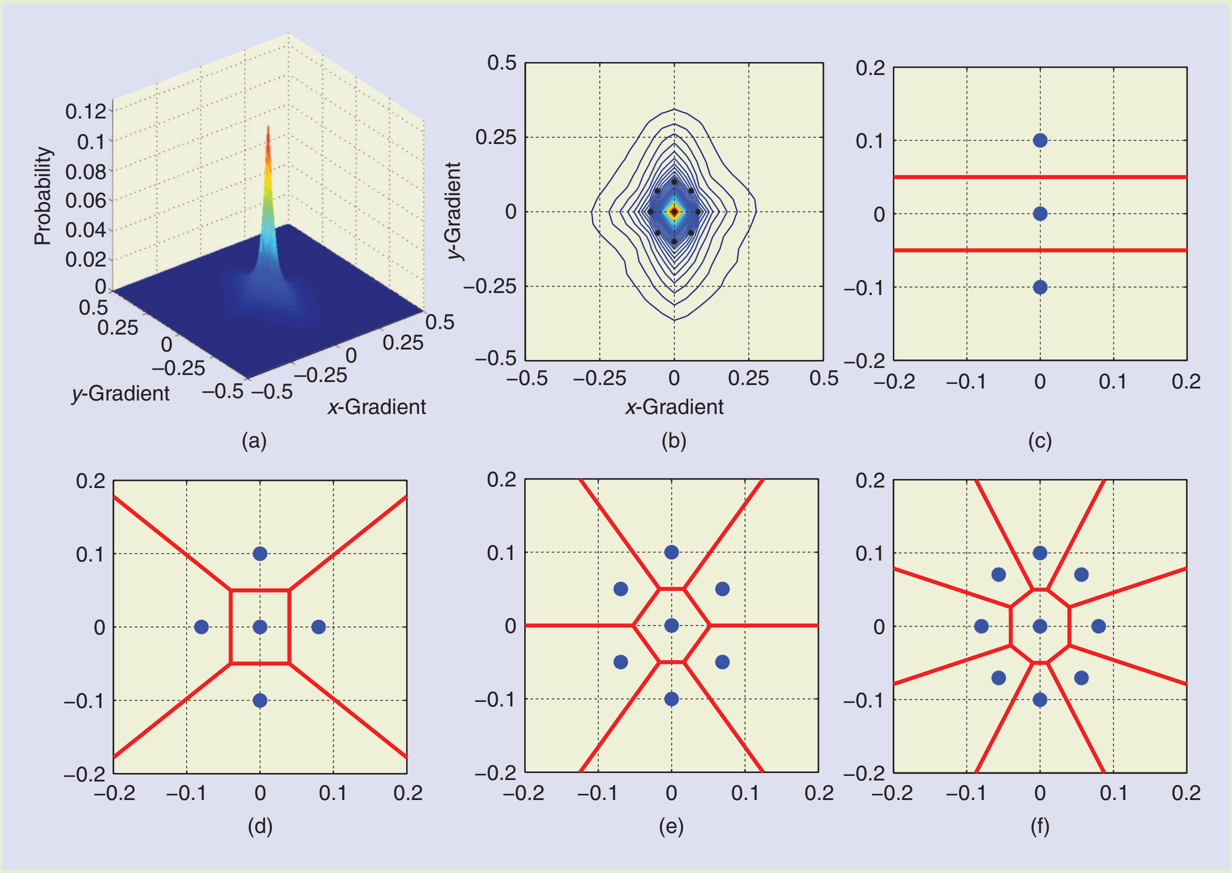
After interest-point detection, we compute a visual word descriptor on a normalized patch. We would like descriptors to be robust to small distortions in scale, orientation, and lighting conditions. Also, we require descriptors to be discriminative, i.e., characteristic of an image or a small set of images. Descriptors that occur in almost every image (the equivalent of the word and in text documents) would not be useful for retrieval. Since the publication of Lowe's article in 1999 [23], the highly discriminative SIFT descriptor remains the most popular descriptor in computer vision. Other examples of feature descriptors are gradient location and orientation histogram (GLOH) by Mikolajczyk and Schmid [22], SURF by Bay et al. [24], and our own compressed histogram of gradients (CHoGs) [25], [26]. Winder and Brown [27], [28] and Mikolajczyk et al. [22] evaluate the performance of the different descriptors.

As a 128-dimensional descriptor, the SIFT descriptor is conventionally stored as 1,024 b (8 b/dimension). However, the size of SIFT descriptor data from an image is typically larger than the size of the JPEG-compressed image itself. Several compression schemes have been proposed to reduce the bit rate of SIFT descriptors. In our recent work [29], we survey the different SIFT compression schemes. They can be broadly categorized into schemes based on hashing [30]–[32], transform coding [29], [33] and vector quantization (VQ) [10], [11], [34]. We note that hashing schemes such as locality-sensitive

CHoG: A LOW BIT-RATE DESCRIPTOR

CHoG builds upon the principles of HoG descriptors with the goal of being highly discriminative at low bit rates. Figure 3 illustrates how the CHoG descriptors are computed.

- The patch is divided into spatial bins, which provides robustness to interest-point localization error. We divide the patch around each interest point into soft log-polar spatial bins using DAISY configurations proposed in [28]. The log-polar configuration has been shown to be more effective than the square-grid configuration used in SIFT [22], [28], [46].
- The joint (d_x, d_y) gradient histogram in each spatial bin is directly captured into the descriptor, as illustrated in Figure S1. CHoG histogram binning exploits the skew in gradient statistics that are observed for patches extracted around interest points.



[FIG S1] The joint (d_x, d_y) gradient distribution (a) over a large number of cells and (b) its contour plot. The greater variance in y axis results from aligning the patches along the most dominant gradient after interest-point detection. (The quantization bin constellations (c) VQ-3, (d) VQ-5, (e) VQ-7, and (f) VQ-9 and their associated Voronoi cells are shown.

hashing (LSH), similarity-sensitive coding (SSC), or spectral hashing (SH) do not perform well at low bit rates. Conventional transform-coding schemes based on principal component analysis (PCA) do not work well because of the highly non-Gaussian statistics of the SIFT descriptor. The VQ schemes based on the product quantizer [34] or a tree-structured vector quantizer [10] are complex and require storage of large codebooks on the mobile device.

In [33], we also explore transform coding of the 64-dimensional SURF descriptor, which also performs poorly at low bit rates. Other popular approaches used to reduce the size of descriptors typically employ dimensionality reduction via PCA or linear discriminant analysis (LDA) [35], [36]. Ke and Sukthankar [35] investigate dimensionality reduction of patches directly via PCA. Hua et al. [36] propose a scheme that uses LDA. Winder and Brown [28] combine the use of PCA with additional optimization of gradient and spatial binning parameters as part of the training step. The disadvantages of PCA and LDA approaches are its high computational complexity and the risk of overtraining for descriptors from a particular data set. Further, with PCA and LDA, descriptors cannot be compared in the compressed domain if entropy

- CHoG retains the information in each spatial bin as a distribution. This allows the use of more effective distance measures such as Kullback Leibler (KL) divergence, and more importantly, allows us to apply quantization and compression schemes that work well for distributions, to produce compact descriptors.

Typically, nine to 13 spatial bins and three to nine gradient bins are chosen, resulting in 27- to 117-dimensional descriptors. For compressing the descriptor, we quantize the gradient histogram in each spatial bin individually. In [25] and [26], we have explored several novel quantization schemes that work well for compressing distributions: quantization by Huffman coding, type coding, and optimal Lloyd-Max VQ. Here, we briefly discuss one of the schemes: type coding, which is linear in complexity to the number of histogram bins and performs close to optimal Lloyd-Max VQ.

Let m represent the number of histogram bins. m varies from three to nine for the CHoG descriptor. Let $P = [p_1, p_2, \dots, p_m] \in R_+^m$ be the original distribution as described by the gradient histogram, and $Q = [q_1, q_2, \dots, q_m] \in R_+^m$ be the quantized probability distribution. First, we construct a lattice of distributions (or types) $Q_n = Q(k_1, \dots, k_m)$ with probabilities

$$q_i = \frac{k_i}{n}, \quad k_i, n \in Z_+, \quad \sum_i k_i = n. \qquad (1)$$

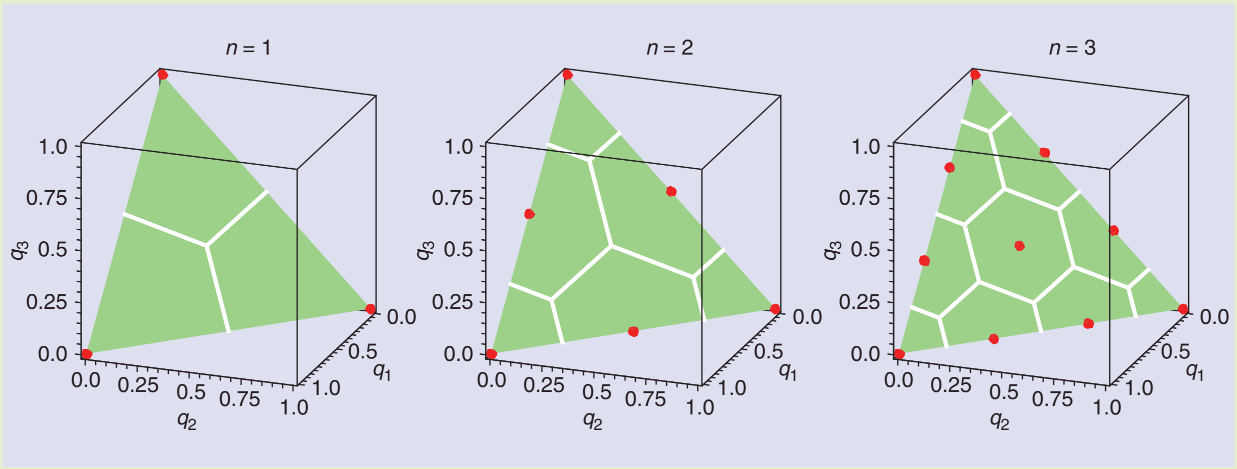
We show several examples of such sets in $m = 3$ dimensions in Figure S2.

The parameter n controls the fidelity of quantization, and higher the value of n parameter, higher the fidelity. Second, after quantizing the distribution P, we compute an index for the type. The total number of types $K(m, n)$ is the number of partitions of n into m terms $k_1 + \cdots + k_m = n$

$$K(m, n) = \binom{n + m - 1}{m - 1}. \qquad (2)$$

The algorithm that maps a type to its index $f_n: \{k_1, \dots, k_m\} \to [0, K(m, n) - 1]$ is described in [26].

Finally, we encode the index in each spatial cell with fixed-length or entropy codes. Fixed-length encoding provides the benefit of compressed domain matching at the cost of a small performance hit. The type quantization and coding scheme described here performs close to optimal Lloyd-Max VQ and does not require storage of codebooks on the mobile client. The CHoG descriptor with type coding at 60 b matches the performance of the 128-dimensional 1,024-b SIFT descriptor [26].



[FIG S2] **Type lattices and their Voronoi partitions in three dimensions** ($m = 3$, $n = 1, 2, 3$).

coding is employed. The 60-b MPEG-7 trace–transform descriptor [37] and binary robust independent elementary features (BRIEFs) [38] are other examples of low bit-rate descriptors proposed in recent literature. Johnson proposes a generalized set of techniques to compress local features in his recent work [39]. Philbin et al. [40] use learning techniques to generate a 32-B descriptor that performs on par with SIFT.

Through our experiments, we came to realize that simply compressing an off-the-shelf descriptor does not lead to the best-rate-constrained image-retrieval performance. One can do better by designing a descriptor with compression in mind. Of course, such a descriptor still has to be robust and highly discriminative. Ideally, it would permit descriptor comparisons in the compressed domain for speedy feature matching. To meet all these requirements simultaneously, we designed the CHoG descriptor [25], [26]. Descriptors based on the distribution of gradients within a patch of pixels have been shown to be highly discriminative [22], [27]. Lowe [15], Bay et al. [24], Dalal and Triggs [41], Freeman and Roth [42], and Winder et al. [28] have proposed histogram of gradient (HoG)-based descriptors.

The CHoG descriptor is designed to work well at low bit rates (see "CHoG: A Low Bit-Rate Descriptor"). CHoG achieves a performance of 1,024-b SIFT at approximately 60 b/descriptor. CHoG is lower dimensional than SIFT because of the more effective gradient and spatial binning schemes. The compact bit representation is a result of lossy quantization and compression schemes that are employed to the descriptor. The CHoG descriptor is a concatenation of distributions (a set of values that sum to one). The problem of lossy quantization of distributions had received little attention in the compression literature prior to the work on CHoG. In our work [25], [26], [43], [44], we explore the problem of quantization and compression of distributions in detail. Further, we propose schemes that map distributions directly to fixed-length codes, which enables matching in the compressed domain. The CHoG descriptor is compared with several other low bit-rate descriptors in the literature in [45].

[CHoG IS LOWER DIMENSIONAL THAN SIFT BECAUSE OF THE MORE EFFECTIVE GRADIENT AND SPATIAL BINNING SCHEMES.]

At 60 b/descriptor, CHoG descriptor data are an order of magnitude smaller than SIFT- or JPEG-compressed images and can be transmitted much faster over slow wireless links. A small descriptor also helps if the database is stored in the mobile device. The smaller the descriptor, the more features can be stored in limited memory.

As illustrated in Figure 3, each interest point has a location, scale, and orientation associated with it. Interest-point locations are needed in the GV step to validate potential candidate matches. The location of each interest point is typically stored as two numbers: x and y coordinates in the image at subpixel accuracy [15]. In a floating point representation, each feature location would require 64 b and 32 b each for x and y. This is comparable in size to the CHoG descriptor itself. We have developed a novel histogram coding scheme to encode the x, y coordinates of feature descriptors [47] (see "Location Histogram Coding"). With location histogram coding (LHC), we can reduce location data by an order of magnitude compared with their floating point representation, without loss in matching accuracy.

A few hundred descriptors per query image are sufficient for achieving high matching accuracy for large databases [14], [26]. Table 1 summarizes data reduction using CHoG and LHC for 500 descriptors per image.

[TABLE 1] DATA REQUIRED TO REPRESENT AN IMAGE FOR MOBILE VISUAL SEARCH.

SCHEME	DATA (kB)
JPEG-COMPRESSED IMAGE	30–40
SIFT + UNCOMPRESSED LOCATION DATA	66.4
CHoG + UNCOMPRESSED LOCATION DATA	7.6
CHoG + COMPRESSED LOCATION DATA	4.0

FEATURE INDEXING AND MATCHING

For a large database of images, comparing the query image against every database image using pairwise feature matching is infeasible. A database with millions of images might contain billions of features. A linear scan through the database would be time consuming for interactive mobile visual search applications. Instead, we must use a data structure that can quickly return a short list of the database candidates most likely to match the query image. The short list may contain false positives as long as the correct match is included. Slower pairwise comparisons can be subsequently performed on just the short list of candidates rather than the entire database.

Many data structures have been proposed for efficiently indexing all the local features in a large image database. Lowe proposes approximate nearest neighbor (ANN) search of SIFT descriptors with a best-bin-first strategy [15]. One of the most popular methods is Sivic and Zisserman's BoF approach [9]. The BoF codebook is trained by k-means clustering of many training descriptors. During a query, scoring the database images can be made fast by using an inverted file index associated with the BoF codebook. To generate a much larger codebook, Nister and Stewenius use hierarchical k-means clustering to create a vocabulary tree (VT) [10]. The VT is explained in greater detail in "VT and Inverted Index." Alternatively, Philbin et al. use randomized k-d trees to partition the feature descriptor space [12]. Subsequent improvements in tree-based quantization and ANN search include greedy N-best paths [49], query expansion [50], efficient updates over time [51], soft binning [12], and Hamming embedding [11].

The problem with hard quantization for k means is that many matching features that should get quantized to the same node end up in different cells due to quantization error. Soft binning schemes proposed in the literature alleviate this problem and improve the matching accuracy. The advantage of the hierarchical scoring approach in [10] is that the soft assignment is given by the structure of the tree and no additional information needs to be stored for each feature. However, the authors in [12] note that the quantization artifacts are not completely removed with hierarchical quantization. To improve performance, Philbin et al. [12] propose a soft binning scheme, where each feature is assigned to n nearest visual words. However, this increases the size of the inverted index by n-fold. The greedy N-best paths scheme [49] also reduces the quantization error but increases the query time, as N best paths need to be explored in the quantization step. Jegou et al. [11] have proposed to combine k-means quantization and binary vector signatures. First, the feature space is divided into a relatively small number of Voronoi cells (20,000) using k means. Each cell is then divided into n independent hyperplanes resulting in 2^n subcells. Jegou et al. suggest that Hamming embedding provides better quantization. However, this is achieved at the expense of higher memory requirements

LOCATION HISTOGRAM CODING

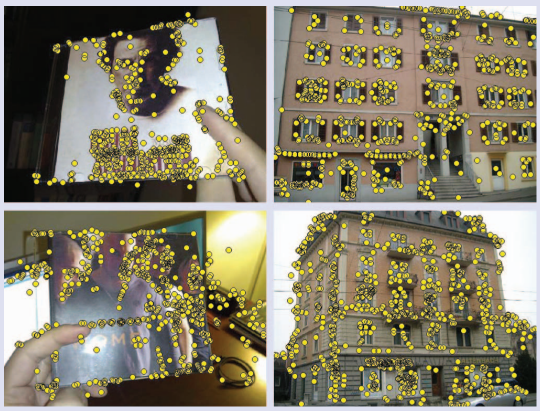
LHC is used to compress feature location data efficiently. We note that the interest points in the images are spatially clustered, as shown in Figure S3.

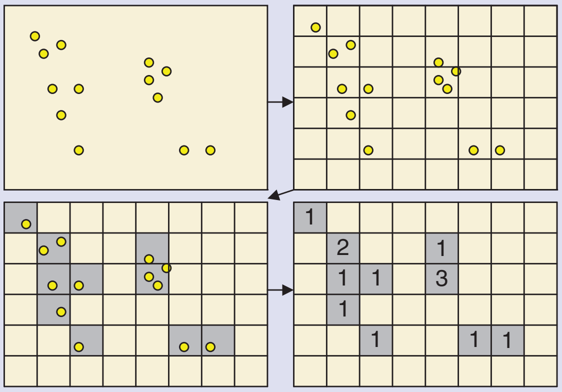
To encode location data, we first generate a two-dimensional (2-D) histogram from the locations of the descriptors (Figure S4). We divide the image into spatial bins and count the number of features within each spatial bin. We compress the binary map, indicating which spatial bins contains features, and a sequence of feature counts, representing the number of features in occupied bins. We encode the binary map using a trained context-based arithmetic coder, with the neighboring bins being used as the context for each spatial bin.

LHC provides two key benefits. First, encoding the locations of a set of N features as the histogram reduces the bit rate by $\log(N!)$ compared to encoding each feature location in sequence [47]. Here, we exploit the fact that the features can be sent in any order. Consider the sample space that represents N features. There are $N!$ number of codes that represent the same feature set because the order does not matter. Thus, if we fix the ordering for the feature set, i.e., using the LHC scheme described earlier, we can achieve a bit savings of $\log(N!)$. For example, for a feature set of 750 features, we achieve a rate savings of $\log(750!)/750 \sim 8$ b/feature.

Second, with LHC, we exploit the spatial correlation between the locations of different descriptors, as illustrated in Figure S3. Different interest-point detectors result in different coding gains. In our experiments, Hessian Laplace has the highest gain followed by SIFT and SURF interest-point detectors. Even if the feature points are uniformly scattered in the image, LHC is still able to exploit the ordering gain, which results in $\log(N!)$ saving in bits.

In our experiments, we found that quantizing the (x, y) location to four-pixel blocks is sufficient for GV. If we use a simple fixed-length coding scheme, then the rate will be $\log(640/4) + \log(640/4) \sim 14$ b/feature for a VGA size image. Using LHC, we can transmit the same location data with ~ 5 b/descriptor; ~ 12.5 times reduction in data compared to a 64-b floating point representation and ~ 2.8 times rate reduction compared to fixed-length coding [48].


[FIG S3] **Interest-point locations in images tend to cluster spatially.**



[FIG S4] **We represent the location of the descriptors using a location histogram. The image is first divided into evenly spaced blocks. We enumerate the features within each spatial block by generating a location histogram.**

and higher query times, as longer inverted files need to be traversed due to the smaller vocabulary size.

As database size increases, the amount of memory used to index the database features can become very large. Thus, developing a memory-efficient indexing structure is a problem of increasing interest. Chum et al. use a set of compact min-hashes to perform near-duplicate image retrieval [52], [53]. Zhang et al. decompose each image's set of features into coarse and refinement signatures [54]. The refinement signature is subsequently indexed by an LSH. Schemes that take advantage of the structure of the database have been proposed recently in [55]–[57]. These schemes are typically applied to databases where there is a lot of redundancy, e.g., each object is represented by images taken from multiple view points. The size of the inverted index is reduced by using geometry to find matching features across images, and only retaining useful features and discarding irrelevant clutter features.

To support the popular VT-scoring framework, inverted index compression methods for both hard-binned and soft-binned VTs have been developed by us [58], as explained in "Inverted Index Compression." The memory for BoF image signatures can be alternatively reduced using the mini-BoF approach [59]. Very recently, visual word residuals on a small BoF codebook have shown promising retrieval results with low memory usage [60], [61]. The residuals are indexed either with PCA and product quantizers [60] or with LSH [61].

VT AND INVERTED INDEX

A VT with an inverted index can be used to quickly compare images in a large database against a query image. If the VT has L levels excluding the root node and each interior node has C children, then a fully balanced VT contains $K = C^L$ leaf nodes. Figure S5 shows a VT with $L = 2$, $C = 3$, and $K = 9$. The VT for a particular database is constructed by performing hierarchical k-means clustering on a set of training feature descriptors representative of the database, as illustrated in Figure S5(a). Initially, C large clusters are generated from all the training descriptors by ordinary k means with an appropriate distance function like L2 norm or symmetric KL divergence. Then, for each large cluster, k-means clustering is applied to the training descriptors assigned to that cluster to generate C smaller clusters. This recursive division of the descriptor space is repeated until there are enough bins to ensure good classification performance. Typically, $L = 6$ and $C = 10$ are selected [10], in which case the VT has $K = 10^6$ leaf nodes.

The inverted index associated with the VT maintains two lists per leaf node, as shown in Figure S5(b). For node k, there is a sorted array of image IDs $\{i_{k1}, i_{k2}, \ldots, i_{kN_k}\}$ indicating which N_k database images have visited that node. Similarly, there is a corresponding array of counts $\{c_{k1}, c_{k2}, \ldots, c_{kN_k}\}$ indicating the frequency of visits. During a query, a database of N total images can be quickly scored by traversing only the nodes visited by the query descriptors. Let $s(i)$ be the similarity score for the ith database image. Initially, prior to visiting any node, $s(i)$ is set to zero. Suppose node k is visited by the query descriptors a total of q_k times. Then, all the images in the inverted list $\{i_{k1}, \ldots, i_{kN_k}\}$ for node k will have their scores incremented according to

$$s(i_{kj}) := s(i_{kj}) + \frac{w_k^2 c_{kj} q_k}{\Sigma_{i_{kj}} \Sigma_q}, \quad j = 1, \ldots, N_k, \tag{3}$$

where w_k is an inverse document frequency (IDF) weight used to penalize often-visited nodes, $\Sigma_{i_{kj}}$ is a normalization factor for database image i_{kj}, and Σ_q is a normalization factor for the query image.

$$w_k = \log(N/N_k), \tag{4}$$

$$\Sigma_{i_{kj}} = \sum_{n=1}^{K} w_n \,(\text{count for DB image } i_{kj} \text{ at node } n), \tag{5}$$

$$\Sigma_q = \sum_{n=1}^{K} w_n \,(\text{count for query image at node } n). \tag{6}$$

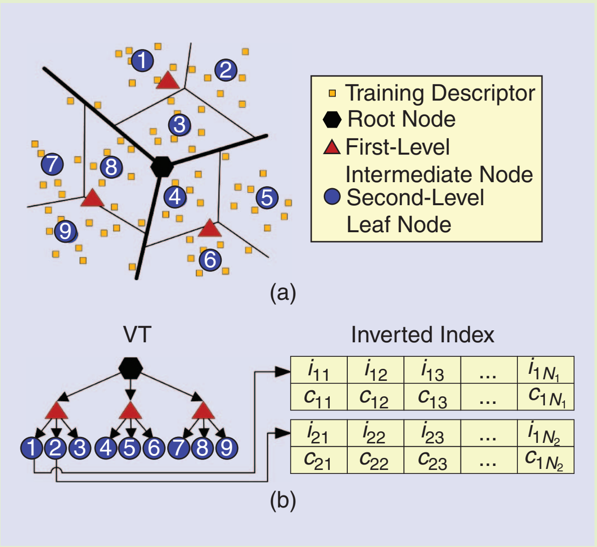
Scores for images at the other nodes visited by the query image are updated similarly. The database images attaining the highest scores $s(i)$ are judged to be the best matching candidates and kept in a short list for further verification.

Soft binning [12] can be used to mitigate the effect of quantization errors for a large VT. As seen in Figure S5(a), some descriptors lie very close to the boundary between two bins. When soft binning is employed, the visit counts are then no longer integers but rather fractional values. For each feature descriptor, the m nearest leaf nodes in the VT are assigned fractional counts

$$c_i = 1/C \cdot \exp\left(-0.5 d_i^2/\sigma^2\right), \quad i = 1, \ldots, m, \tag{7}$$

$$C = \sum_{i=1}^{m} \exp\left(-0.5 d_i^2/\sigma^2\right), \tag{8}$$

where d_i is the distance between the ith closest leaf node and the feature descriptor, and S is appropriately chosen to maximize classification accuracy.


1
2
3
4
5
6
7
8
9
Training Descriptor
Root Node
First-Level Intermediate Node
Second-Level Leaf Node
(a)
VT
Inverted Index
i_{11} i_{12} i_{13} ... i_{1N_1}
c_{11} c_{12} c_{13} ... c_{1N_1}
i_{21} i_{22} i_{23} ... i_{1N_2}
c_{21} c_{22} c_{23} ... c_{1N_2}
(b)

[FIG S5] **(a) Construction of a VT by hierarchical *k*-means clustering of training feature descriptors. (b) VT and the associated inverted index.**

GEOMETRIC VERIFICATION

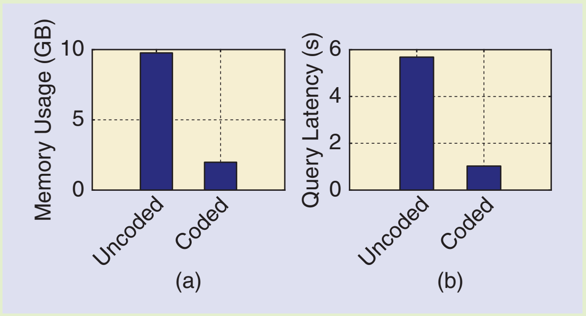
The GV typically follows the feature matching step. In this stage, we use location information of query and database features to confirm that the feature matches are consistent with a change in viewpoint between the two images. We perform pairwise matching of feature descriptors and evaluate geometric consistency of correspondences as shown in Figure 4. The geometric transform between query and database image is estimated using robust regression techniques such as RANSAC [66] or the Hough transform [15]. The transformation can be represented by the fundamental matrix that incorporates three-dimensional (3-D) geometry, or simpler homography or affine models. The GV tends to be computationally expensive, which limits the list of candidate images to a small number.

A number of groups have investigated ways to speed up the GV process. Semilocal geometric constraints have been proposed in [9], [13], [67], and [68] to either filter out or propose feature matches. The authors Jegou et al.

INVERTED INDEX COMPRESSION

For a database containing 1 million images and a VT that uses soft binning, each image ID can be stored in a 32-b unsigned integer, and each fractional count can be stored in a 32-b float in the inverted index. The memory usage of the entire inverted index is $\sum_{k=1}^{k} N_k \cdot 64$ bits, where N_k is the length of the inverted list at the kth leaf node. For a database of 1 million product images, this amount of memory reaches 10 GB, a huge amount for even a modern server. Such a large memory footprint limits the ability to run other concurrent processes on the same server, such as recognition systems for other databases. When the inverted index's memory usage exceeds the server's available random access memory (RAM), swapping between main and virtual memory occurs, which significantly slows down all processes.

A compressed inverted index [58] can significantly reduce memory usage without affecting recognition accuracy. First, because each list of IDs $\{i_{k1}, i_{k2}, \ldots, i_{kN_k}\}$ is sorted, it is more efficient to store consecutive ID differences $\{d_{k1} = i_{k1}, d_{k2} = i_{k2} - i_{k1}, \ldots, d_{kN_k} = i_{kN_k} - i_{k(N_k-1)}\}$ in place of the IDs. This practice is also commonly used in text retrieval [62]. Second, the fractional visit counts can be quantized to a few representative values using Lloyd-Max quantization. Third, the distributions of the ID differences and visit counts are far from uniform, so variable-length coding can be much more rate efficient than fixed-length coding. Using the distributions of the ID differences and visit counts, each inverted list can be encoded using an arithmetic code (AC) [63]. Since keeping the decoding delay low is very important for interactive mobile visual search applications, a scheme that allows ultra-fast decoding is often preferred over AC. The carryover code [64] and recursive bottom-up complete (RBUC) code [65] have been shown to be at least ten times faster in decoding than AC, while achieving comparable compression gains as AC. The carryover and RBUC codes attain these speedups by enforcing word-aligned memory accesses.

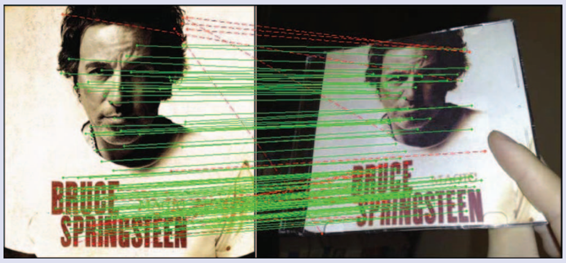


[FIG S6] (a) Memory usage for inverted index with and without compression. A five times savings in memory is achieved with compression. (b) Server-side query latency (per image) with and without compression. The RBUC code is used to encode the inverted index.

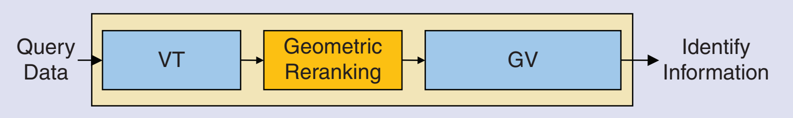
Figure S6(a) compares the memory usage of the inverted index with and without compression using the RBUC code. Index compression reduces memory usage from nearly 10 GB to 2 GB. This five times reduction leads to a substantial speedup in server-side processing, as shown in Figure S6(b). Without compression, the large inverted index causes swapping between main and virtual memory and slows down the retrieval engine. After compression, memory swapping is avoided and memory congestion delays no longer contribute to the query latency.

[11] use weak geometric consistency checks to rerank images based on the orientation and scale information of all features. The authors in [53] and [69] propose incorporating geometric information into the VT matching or hashing step. In [70] and [71], the authors investigate how to speed up RANSAC estimation itself. Philbin et al. [72] use single pairs of matching features to propose hypotheses of the geometric transformation model and verify only possible sets of hypotheses. Weak geometric consistency checks are typically used to rerank a larger candidate list of images, before a full GV is performed on a shorter candidate list.

To speed up GV, one can add a geometric reranking step before the RANSAC GV step, as illustrated in Figure 5. In [73], we propose a reranking step that incorporates geometric information directly into the fast index lookup stage and use it to reorder the list of top matching images (see "Fast Geometric Reranking"). The main advantage of the scheme is that it only requires x, y feature location data and does not use scale



[FIG4] In the GV step, we match feature descriptors pairwise and find feature correspondences that are consistent with a geometric model. True feature matches are shown in red. False feature matches are shown in green.

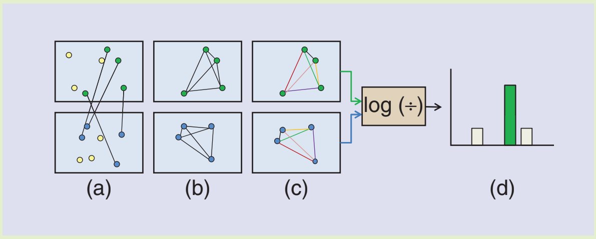


[FIG5] An image retrieval pipeline can be greatly sped up by incorporating a geometric reranking stage.

FAST GEOMETRIC RERANKING

We have proposed a fast geometric reranking algorithm in [73] that uses *x, y* locations of features to rerank a short list of candidate images. First, we generate a set of potential feature matches between each query and database image based on VT-quantization results. After generating a set of feature correspondences, we calculate a geometric score between them. The process used to compute the geometric similarity score is illustrated in Figure S7. We find the distance between the two features in the query image and the distance between the corresponding matching features in the database image. The ratio of the distance corresponds to the scale difference between the two images. We repeat the ratio calculation for features in the query image that have matching database features. If there exists a consistent set of ratios (as indicated by a peak in the histogram of distance ratios), it is more likely that the query image and the database image match.

The geometric reranking is fast because we use the VT-quantization results directly to find potential feature matches and use a really simple similarity scoring scheme.



[FIG S7] The location geometric score is computed as follows: (a) features of two images are matched based on VT quantization, (b) distances between pairs of features within an image are calculated, (c) log-distance ratios of the corresponding pairs (denoted by color) are calculated, and (d) histogram of log-distance ratios is computed. The maximum value of the histogram is the geometric similarity score. A peak in the histogram indicates a similarity transform between the query and database image.

The time required to calculate a geometric similarity score is one to two orders of magnitude less than using RANSAC. Typically, we perform fast geometric reranking on the top 500 images and RANSAC on the top 50 ranked images.

or orientation information as in [11]. As scale and orientation data are not used, they need not be transmitted by the client, which reduces the amount of data transferred. We typically run fast geometric reranking on a large set of candidate database images and reduce the list of images that we run RANSAC on.

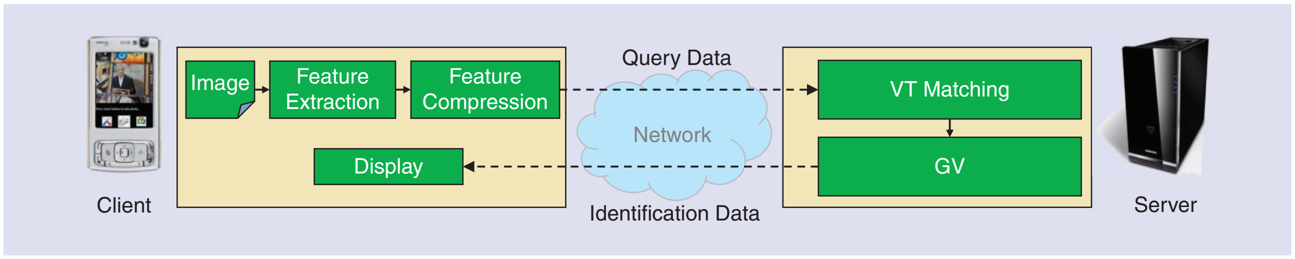
Before discussing system performance results, we provide a list of important references for each module in the matching pipeline in Table 2.

SYSTEM PERFORMANCE

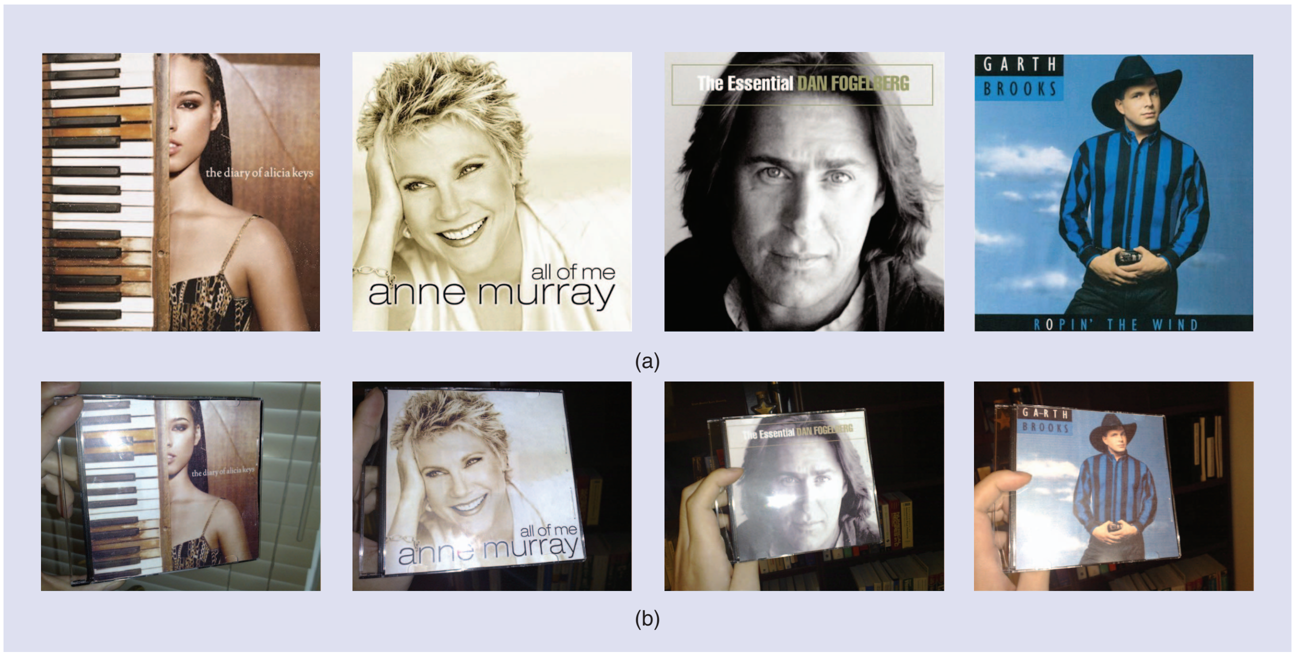
What performance can we expect for a mobile visual search system that incorporates all the ideas discussed so far? To answer this question, we have a closer look at the experimental Stanford Product Search System (Figure 6). For evaluation, we use a database of 1 million CD, digital versatile disk (DVD), and book cover images, and a set of 1,000 query images (500 $\times$ 500 pixel resolution) [75] exhibiting challenging photometric and geometric distortions, as shown in Figure 7. For

[TABLE 2] SUMMARY OF REFERENCES FOR MODULES IN A MATCHING PIPELINE.

MODULE	LIST OF REFERENCES
FEATURE EXTRACTION	HARRIS AND STEPHENS [17], LOWE [15], [23], MATAS ET AL. [18], MIKOLAJCZYK ET AL. [16], [22], DALAL AND TRIGGS [41], ROSTEN AND DRUMMOND [19], BAY ET AL. [20], WINDER ET AL. [27], [28], CHANDRASEKHAR ET AL. [25], [26], PHILBIN ET AL. [40]
FEATURE INDEXING AND MATCHING	SCHMID AND MOHR [13], LOWE [15], [23], SIVIC AND ZISSERMAN [9], NISTÉR AND STEWÉNIUS [10], CHUM ET AL. [50], [52], [53], YEH ET AL. [51], PHILBIN ET AL. [12], JEGOU ET AL. [11], [59], [60], ZHANG ET AL. [54] CHEN ET AL. [58], PERRONNIN [61], MIKULIK ET AL. [55], TURCOT AND LOWE [56], LI ET AL. [57]
GV	FISCHLER AND BOLLES [66], SCHAFFALITZKY AND ZISSERMAN [74], LOWE [15], [23], CHUM ET AL. [53], [70], [71] FERRARI ET AL. [68], JEGOU ET AL. [11], WU ET AL. [69], TSAI ET AL. [73]



[FIG6] Stanford Product Search system. Because of the large database, the image-recognition server is placed at a remote location. In most systems [1], [3], [7], 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.



[FIG7] Example image pairs from the data set. (a) A clean database picture is matched against (b) a real-world picture with various distortions.

the client, we use a Nokia 5800 mobile phone with a 300-MHz central processing unit (CPU). For the recognition server, we use a Linux server with a Xeon E5410 2.33-GHz CPU and 32 GB of RAM. We report results for both 3G and wireless local area network (WLAN) networks. For 3G, experiments are conducted in an AT&T 3G wireless network, averaged over several days, with a total of more than 5,000 transmissions at indoor locations where such an image-based retrieval system would be typically used.

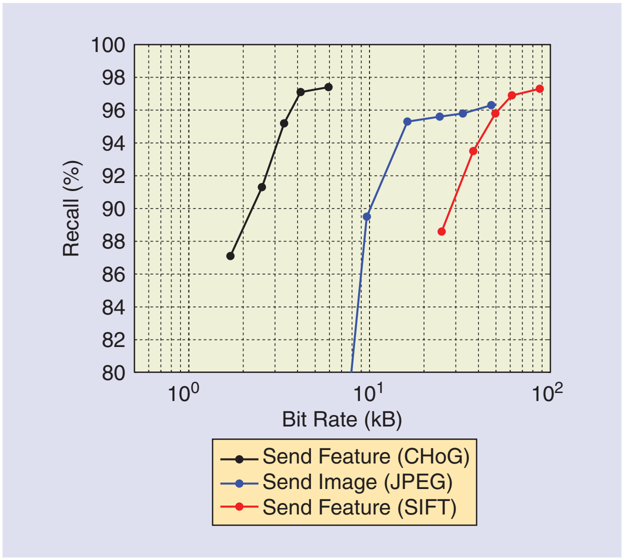
We evaluate two different modes of operation. In send features mode, we process the query image on the phone and transmit compressed query features to the server. In send image mode, we transmit the query image to the server, and all operations are performed on the server.

We discuss results of the three key aspects that are critical for mobile visual search applications: retrieval accuracy, system latency, and power. A recurring theme throughout this section will be the benefits of performing feature extraction on the mobile device compared to performing all processing on a remote server.

RETRIEVAL ACCURACY

It is relatively easy to achieve high precision (low false positives) for mobile visual search applications. By requiring a minimum number of feature matches after RANSAC GV, we can avoid false positives entirely. To avoid false-positive matches, we set a minimum number of matching features after RANSAC GV to 12, which is high enough to avoid false positives. We define recall as the percentage of query images correctly retrieved. Our goal then is to maximize recall at a negligibly low false-positive rate.

Figure 8 compares the recall for the three schemes: send features (CHoG), send features (SIFT), and send image (JPEG) at precision one. For the JPEG scheme, the bit rate is varied by changing the quality of compression. For the SIFT scheme, we extract the SIFT descriptors on the mobile device and transmit each descriptor uncompressed as 1,024 b. For the CHoG scheme, we need to transmit about 60 b/descriptor across the network. For SIFT and CHoG schemes, we sweep the recall-bit rate curve by varying the number of descriptors transmitted. For a given budget of



[FIG8] Bit-rate comparisons of different schemes. CHoG descriptor data are an order of magnitude smaller compared to the JPEG images or uncompressed SIFT descriptors.

features, the descriptors with the highest Hessian response are transmitted. The descriptors are transmitted in the order imposed by the LHC scheme discussed in "Location Histogram Coding."

First, we observe that a recall of 96% is achieved at the highest bit rate for challenging query images even with a million images in the database. Second, we observe that the performance of the JPEG scheme rapidly deteriorates at low bit rates. The performance suffers at low bit rates as the interest-point detection fails due to JPEG-compression artifacts. Third, we note that transmitting uncompressed SIFT data is almost always more expensive than transmitting JPEG-compressed images. Finally, we observe that the amount of data for CHoG descriptors is an order of magnitude smaller than JPEG images or SIFT descriptors at the same retrieval accuracy.

THE SYSTEM LATENCY CAN BE BROKEN DOWN INTO THREE COMPONENTS: PROCESSING DELAY ON CLIENT, TRANSMISSION DELAY, AND PROCESSING DELAY ON SERVER.

SYSTEM LATENCY

The system latency can be broken down into three components: processing delay on client, transmission delay, and processing delay on server.

CLIENT- AND SERVER-PROCESSING DELAY

We show the time taken for the different operations on the client and server in Table 3. The send features mode requires approximately 1 s for feature extraction on the client. However, this increase in client-processing time is more than compensated by the decrease in transmission latency, compared to send image, as illustrated in Figures 9 and 10. On the server, using VT matching with a compressed inverted index, we can search through a million image database in 100 ms. We perform GV on a short list of 50 candidates after fast geometric reranking of the top 500 candidate images. We can achieve <1 s server processing latency while maintaining high recall.

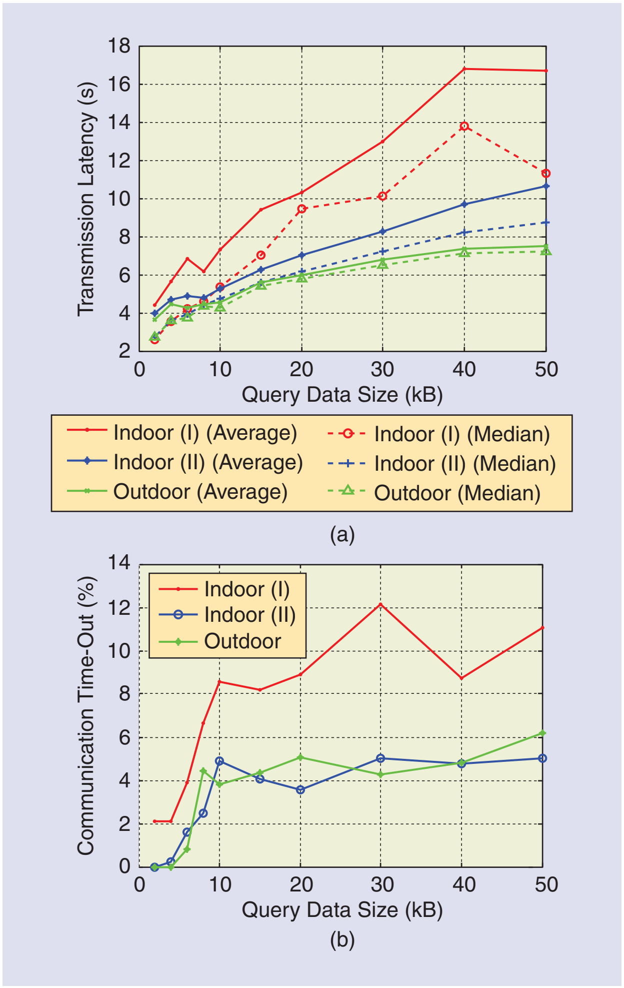
TRANSMISSION DELAY

The transmission delay depends on the type of network used. In Figure 10, we observe that the data transmission time is insignificant for a WLAN network because of the high bandwidth available. However, the transmission time turns out to be a bottleneck for 3G networks. In Figure 9, we present the experimental results for sending data over a 3G wireless network. We vary query data sizes from that of typical compressed query features (3–4 kB) to typical JPEG query images (50 kB) to learn how query size affects transmission time. The communication time-out was set to 60 s. We have conducted the experiment continuously for several days. We tested at three different locations, typical locations where a user might use the visual search application.

The median and average transmission latency of our experiments are shown in Figure 9. Sending the compressed query features typically takes 3–4 s. The time required to send the compressed query image is several times longer and varies significantly at different locations. However, the transmission delay does not include the cases when the communication

[TABLE 3] PROCESSING TIMES.

CLIENT-SIDE OPERATIONS	TIME (s)
IMAGE CAPTURE	1–2
FEATURE EXTRACTION AND COMPRESSION *(FOR SEND IMAGE MODE)*	1–1.5
SERVER-SIDE OPERATIONS	**TIME (ms)**
FEATURE EXTRACTION *(FOR SEND IMAGE MODE)*	100
VT MATCHING	100
FAST GEOMETRIC RERANKING (PER IMAGE)	0.46
GV (PER IMAGE)	30



[FIG9] Measured transmission latency (a) and time-out percentage (b) for transmitting queries of different size over a 3G network. Indoor (I) is tested indoors with poor connectivity. Indoor (II) is tested indoors with good reception. Outdoor is tested outside of buildings.

fails entirely, which increases with the query size. We show the percentage of transmissions that experience a time-out in Figure 9(b). The time-out percentage of transmitting the compressed query features is much lower than that of transmitting compressed query images because of their smaller query size.

END-TO-END LATENCY

We compare end-to-end latency for the different schemes in Figure 10. For WLAN, we observe that <1 s query latency is achieved for the send image mode. The send features mode is slower because of the processing delay on the client. With such fast response times over WLAN, we are able to operate our system in a continuous mobile augmented reality mode [76].

For 3G networks, network latency remains the bottleneck as seen in Figure 10. In this scenario, there is a significant benefit in sending compressed features. The send features mode reduces the system latency by two times compared with the send image mode.

ENERGY CONSUMPTION

On a mobile device, we are constrained by the energy of the battery, and hence, conserving the energy is critical. We measure the average energy consumption associated with a single query using the Nokia Energy Profiler (http://store.ovi.com/content/17374) on the Nokia 5800 phone.

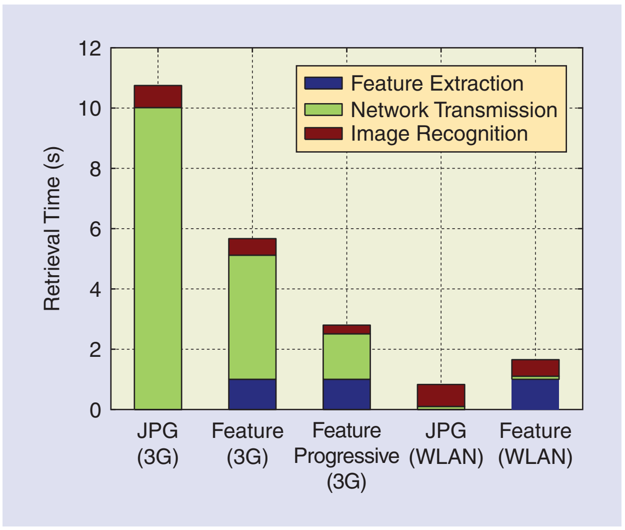
We show the average energy consumption for a single query using send features and send image for WLAN and 3G network connections in Figure 11. For 3G connections, the energy consumed in the send image mode is almost three times as much as send features. The additional time needed to transmit image data compared with feature data results in a greater amount of energy being consumed. For WLAN transmission, send image consumes less energy, since feature extraction on the mobile client is not required.

[FEATURE COMPRESSION IS A KEY PROBLEM FOR VISUAL SEARCH TO WORK WITH MOBILE DEVICES.]

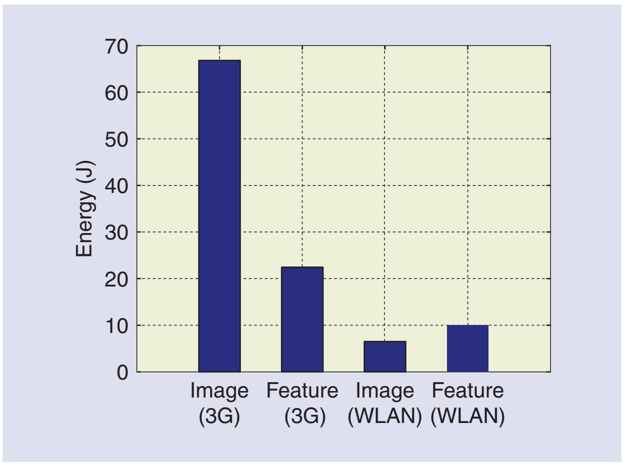
Finally, we compute the number of image queries the mobile can send before the battery runs out of power. A typical phone battery has voltage of 3.7 V and a capacity of ~1,000 mAh (or ~13.3 kJ). Hence, for 3G connections, the maximum number of images that the mobile can send is 13.3 kJ/70 J = ~190 total queries. For send features, we would be able to perform 13.3 kJ/21 J = ~630 total queries, which is three times as many queries as the send image can perform. This difference becomes even more important as we move toward streaming augmented reality applications.

CONCLUDING REMARKS

Mobile visual search is ready for prime time. State-of-the-art systems today achieve more than 95% recall at negligible false-positive rate for databases with more than one million classes, a recognition performance that is sufficient for many applications. The key is robust, and discriminative local features are used in a bag-of-visual-words approach. Robustness against scale changes and rotation is achieved by interest-point detection algorithms that robustly yield not only a location but also feature-scale and dominant orientation. This permits feature descriptors computed on canonical patches in a local coordinate system. Robustness against brightness and contrast variations is achieved by normalizing the patch variance and using only the image gradient to compute feature descriptors. This takes care of much of the variability among corresponding query and database features, but not all. Foreshortening affects the feature descriptor, and correspondences can no longer be established if the angle becomes extreme. Adding foreshortened



[FIG10] An end-to-end latency for different schemes. Compared to send image scheme, we achieve approximately four times reduction in average system latency using progressive transmission of CHoG feature descriptors in a 3G network.



[FIG11] Average energy consumption of a single query using send image and send features mode for various types of transmission.

descriptors to the database or using affine invariant descriptors are possible remedies. Robustness against blur (particularly motion blur) is typically lacking but would be highly desirable. By using clever hierarchical data structures such as VTs, fast retrieval can be achieved in less than 1 s on a typical desktop CPU for databases of more than one million images. Such speed requires a precomputed inverted file index that is stored entirely in RAM, and seeking data on a hard disk would slow down the retrieval considerably. Inverted index compression can be used to fit even larger data structures in RAM. For large databases, VTs must be combined with a GV stage to avoid false positives. Geometric consistency checks are computationally expensive, so it makes sense to use simple geometric reranking to reduce the number of candidate images.

[STATE-OF-THE-ART TECHNOLOGY WORKS BEST ON HIGHLY TEXTURED RIGID PLANAR OBJECTS UNDER CONTROLLED LIGHTING CONDITIONS.]

Feature compression is a key problem for visual search to work with mobile devices. Sending a JPEG image as a query over a slow wireless link can take a longer time; it is better to extract the salient features on the phone instead and send these features as the query to the server. Note that this approach also provides a certain degree of privacy. For a small database, compressed database features could be stored on the mobile device, and matching could be performed directly on the device without wireless communication. One can compress well-known feature descriptors, such as SIFT, trading off recognition performance against size. Better performance, however, is achieved by the CHoG descriptor that was directly designed for rate-constrained recognition. CHoG needs about 60 b/feature, including its location.

As an example throughout, we have used the Stanford Product Search system that implements many of the ideas discussed. We observe that the processing latency on the client and server is typically on the order of approximately 1 s. The transmission delay depends on the network being used. For WLAN, the transmission delay is insignificant, and transmitting a JPEG image as a query is faster than extracting CHoG features on the phone. For 3G networks, the transmission delay is the bottleneck in an end-to-end system latency. By transmitting feature data, we can reduce end-to-end system latency to 2–3 s, compared to sending the image that takes several times longer. Somewhat counter to intuition, extracting and transmitting feature data on the mobile client requires three times less energy than sending the image.

In the next few years, we can expect a broad range of mobile visual search applications in domains ranging from artwork, texts, books, CD covers, and buildings. State-of-the-art technology works best on highly textured rigid planar objects under controlled lighting conditions. The problem becomes much more challenging for 3-D objects (e.g., buildings), especially when the lighting conditions of the query image are drastically different from the reference database image. More generic object categories like plants, furniture, and apparel are challenging for recognition too. Text detection and optical character recognition are challenging for blurry low-quality camera phone images. However, this is becoming less of a problem as smartphones become pervasive.

Numerous open problems remain. Accurate and near-instantaneous Web-scale visual search with billions of images will likely remain as one of the grand challenges of multimedia technology for the years to come. Also, we would like to perform mobile visual search at video rates without ever pressing a button. Although faster processors and networks will get us closer to this goal, lower-complexity image-analysis algorithms are urgently needed. Hardware support on mobile devices should also help. It is envisioned that an ongoing standardization activity in MPEG on compact descriptors for visual search will enable interoperability between databases and applications, enable hardware support on mobile devices, and reduce load on wireless networks carrying visual search related data. Ultimately, we may expect to see ubiquitous mobile augmented reality systems that continuously superimpose information and links on everything the camera of a mobile device sees, thus seamlessly linking the virtual world and the physical world.

AUTHORS

Bernd Girod (bgirod@stanford.edu) received his engineering doctorate from the University of Hannover, Germany, and his M.S. degree from Georgia Institute of Technology. He was a professor in the Electrical Engineering Department of the University of Erlangen-Nuremberg. He has been a professor of electrical engineering and computer science in the Information Systems Laboratory of Stanford University, California, since 1999. He has published more than 450 conference and journal papers, as well as five books. He received the EURASIP Signal Processing Best Paper Award in 2002, the IEEE Multimedia Communication Best Paper Award in 2007, the EURASIP Image Communication Best Paper Award in 2008, as well as the EURASIP Technical Achievement Award in 2004. He has been involved with several start-up ventures, such as Polycom, Vivo Software, 8 × 8, and RealNetworks. He is a Fellow of the IEEE, a member of the German National Academy of Sciences (Leopoldina), and a fellow of EURASIP. His current research interests include video compression, networked media systems, and image-based retrieval.

Vijay Chandrasekhar (vijayc@stanford.edu) received his M.S. and B.S. degrees in electrical and computer engineering from Carnegie Mellon University. He is a Ph.D. student in the Image, Video, and Multimedia Systems (IVMS) group in the Department of Electrical Engineering at Stanford University. He is a Member of the IEEE. His research focuses on content-based image retrieval with a focus on low bitrate image retrieval for mobile visual search applications.

David M. Chen (dmchen@stanford.edu) received his M.S. and B.S. degrees in electrical engineering from Stanford

University. He is a Ph.D. student in the IVMS group in the Department of Electrical Engineering at Stanford University. He received a departmental Outstanding Teaching Assistant Award for the graduate-level digital image processing class taught at Stanford, in which he helped to integrate mobile image processing on android into the curriculum. He is a Member of the IEEE. His research interests include large-scale content-based image and video retrieval for mobile visual search applications.

Ngai-Man Cheung (ncheung@stanford.edu) received his Ph.D. degree from the University of Southern California (USC), Los Angeles, in 2008. He is currently a postdoctoral researcher with the Information Systems Laboratory at Stanford University. He is a Member of the IEEE. His research interests include multimedia signal processing, compression, and retrieval.

Radek Grzeszczuk (radek.grzeszczuk@nokia.com) received his Ph.D. degree in computer science from the University of Toronto. He is a principal scientist at Nokia Research Center, Palo Alto. He is a Member of the IEEE. His research interests include mobile visual search, computer graphics, and augmented reality.

Yuriy Reznik (yreznik@qualcomm.com) received his Ph.D. degree in computer science from Kiev University. He was a coinventor of RealAudio and RealVideo algorithms at RealNetworks, Inc. (Seattle, Washington, 1998–2005) and a contributor to several speech-, audio-, and video-coding standards, including ITU-T H.264/MPEG-4 AVC, MPEG-4 ALS, MPEG-C parts 1,2, and ITU-T G.718. He is currently a staff engineer at Qualcomm Inc., in San Diego, California, and cochair of an AdHoc Group on Visual Search in MPEG. He is a Senior Member of the IEEE and a coauthor of more than 50 academic papers and more than 30 (including nine granted) U.S. patents.

Gabriel Takacs (gtakacs@stanford.edu) began his academic career at Harvey Mudd College, where he studied systems engineering before starting graduate school at Stanford University. At Stanford, he has been working on mobile augmented reality and visual search. He is a Member of the IEEE.

Sam S. Tsai (sstsai@stanford.edu) is a Ph.D candidate in the Department of Electrical Engineering in Stanford University. He is a Member of the IEEE. His research interests include image and video compression, mobile multimedia systems, and mobile image retrieval.

Ramakrishna Vedantham (Ramakrishna.Vedantham@nokia.com) received his M.S. degree in electrical engineering in 2000 from the University of Texas at Dallas and B.Tech. degree in electrical and computer engineering in 1996 from the National Institute of Technology, Calicut, India. He is a senior researcher at Nokia Research Center, Palo Alto. He is a Member of the IEEE. His current research interests include mobile augmented reality systems, content-based image retrieval, and visual navigation.

REFERENCES

[1] Google. (2009). Google Goggles [Online]. Available: http://www.google.com/mobile/goggles/

[2] Nokia. (2006). Nokia Point and Find [Online]. Available: http://www.pointandfind.nokia.com

[3] Kooaba. (2007). Kooaba [Online]. Available: http://www.kooaba.com

[4] B. Erol, E. Antúnez, and J. Hull, "Hotpaper: Multimedia interaction with paper using mobile phones," in *Proc. 16th ACM Multimedia Conf.*, New York, 2008.

[5] J. Graham and J. J. Hull, "Icandy: A tangible user interface for itunes," in *Proc. CHI '08: Extended Abstracts on Human Factors in Computing Systems*, Florence, Italy, 2008.

[6] J. J. Hull, B. Erol, J. Graham, Q. Ke, H. Kishi, J. Moraleda, and D. G. Van Olst, "Paper-based augmented reality," in *Proc. 17th Int. Conf. Artificial Reality and Telexistence (ICAT)*, Washington, DC, 2007.

[7] Amazon. (2007). SnapTell [Online]. Available: http://www.snaptell.com

[8] G. Takacs, V. Chandrasekhar, N. Gelfand, Y. Xiong, W. Chen, T. Bismpigiannis, R. Grzeszczuk, K. Pulli, and B. Girod, "Outdoors augmented reality on mobile phone using loxel-based visual feature organization," in *Proc. ACM Int. Conf. Multimedia Information Retrieval (ACM MIR)*, Vancouver, Canada, Oct. 2008.

[9] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, Washington, DC, 2003.

[10] D. Nistér and H. Stewénius, "Scalable recognition with a vocabulary tree," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, New York, June 2006.

[11] H. Jegou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search," in *Proc. European Conf. Computer Vision (ECCV)*, Berlin, Heidelberg, 2008.

[12] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Lost in quantization—Improving particular object retrieval in large scale image databases," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Anchorage, AK, June 2008.

[13] C. Schmid and R. Mohr, "Local grayvalue invariants for image retrieval," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, pp. 530–535, 1997.

[14] S. S. Tsai, D. M. Chen, V. Chandrasekhar, G. Takacs, N. M. Cheung, R. Vedantham, R. Grzeszczuk, and B. Girod, "Mobile product recognition," in *Proc. ACM Multimedia (ACM MM)*, Florence, Italy, Oct. 2010.

[15] D. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[16] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool, "A comparison of affine region detectors," *Int. J. Comput. Vis.*, vol. 65, no. 1-2, pp. 43–72, 2005.

[17] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. 4th Alvey Vision Conf.*, 1988.

[18] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide baseline stereo from maximally stable extremal regions," in *Proc. British Machine Vision Conf. (BMVC)*, Cardiff, Wales, Sept. 2002.

[19] E. Rosten and T. Drummond, "Machine learning for high speed corner detection," in *Proc. European Conf. Computer Vision (ECCV)*, Graz, Austria, May 2006.

[20] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *Proc. European Conf. Computer Vision (ECCV)*, Graz, Austria, May 2006.

[21] G. Takacs, V. Chandrasekhar, D. M. Chen, S. S. Tsai, R. Grzeszczuk, and B. Girod, "Unified real-time tracking and recognition with rotation invariant fast features," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, SFO, CA, to be published.

[22] K. Mikolajczyk and C. Schmid, "Performance evaluation of local descriptors," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 27, no. 10, pp. 1615–1630, 2005.

[23] D. Lowe, "Object recognition from local scale-invariant features," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, Aug. 1999.

[24] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust feature," *Comput. Vis. Image Understand.*, vol. 110, no. 3, pp. 346–359, 2008.

[25] V. Chandrasekhar, G. Takacs, D. M. Chen, S. S. Tsai, R. Grzeszczuk, and B. Girod, "CHoG: Compressed histogram of gradients—A low bit rate feature descriptor," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Miami, FL, June 2009.

[26] V. Chandrasekhar, Y. Reznik, G. Takacs, D. M. Chen, S. S. Tsai, R. Grzeszczuk, and B. Girod, "Study of quantization schemes for low bitrate CHoG descriptors," in *Proc. IEEE Int. Workshop Mobile Vision (IWMV)*, San Francisco, CA, June 2010.

[27] S. Winder and M. Brown, "Learning local image descriptors," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Minneapolis, MN, 2007, pp. 1–8.

[28] S. Winder, G. Hua, and M. Brown, "Picking the best daisy," in *Proc. Computer Vision and Pattern Recognition (CVPR)*, Miami, FL, June 2009.

[29] V. Chandrasekhar, M. Makar, G. Takacs, D. M. Chen, S. S. Tsai, N. M. Cheung, R. Grzeszczuk, Y. Reznik, and B. Girod, "Survey of SIFT compression schemes," in *Proc. Int. Mobile Multimedia Workshop (IMMW), IEEE Int. Conf. Pattern Recognition (ICPR)*, Istanbul, Turkey, Aug. 2010.

[30] C. Yeo, P. Ahammad, and K. Ramchandran, "Rate-efficient visual correspondences using random projections," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, San Diego, CA, Oct. 2008.

[31] A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large image databases for recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Anchorage, AK, 2008.

[32] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. Neural Information Processing Systems (NIPS)*, Vancouver, BC, Canada, Dec. 2008.

[33] V. Chandrasekhar, G. Takacs, D. M. Chen, S. S. Tsai, and B. Girod, "Transform coding of feature descriptors," in *Proc. Visual Communications and Image Processing Conf. (VCIP)*, San Jose, CA, Jan. 2009.

[34] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. Pattern Anal. Machine Intell.*, 2010, to be published.

[35] Y. Ke and R. Sukthankar, "PCA-SIFT: A more distinctive representation for local image descriptors," in *Proc. Conf. Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2004, vol. 2, pp. 506–513.

[36] G. Hua, M. Brown, and S. Winder, "Discriminant embedding for local image descriptors," in *Proc. Int. Conf. Computer Vision (ICCV)*, Rio de Janeiro, Brazil, 2007.

[37] P. Brasnett and M. Z. Bober, "Robust visual identifier using the trace transform," in *Proc. IET Visual Information Engineering Conf. (VIE)*, London, U.K., July 2007.

[38] M. Calonder, V. Lepetit, and P. Fua, "Brief: Binary robust independent elementary features," in *Proc. European Conf. Computer Vision (ECCV)*, Crete, Greece, Oct. 2010.

[39] M. Johnson, "Generalized descriptor compression for storage and matching," in *Proc. British Machine Vision Conf. (BMVC)*, London, U.K., June 2010.

[40] J. Philbin, M. Ishard, J. Sivic, and A. Zisserman, "Descriptor learning for efficient retrieval," in *Proc. European Conf. Computer Vision (ECCV)*, Crete, Greece, Sept. 2010.

[41] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, San Diego, CA, June 2005.

[42] W. T. Freeman and M. Roth, "Orientation histograms for hand gesture recognition," in *Proc. Int. Workshop Automatic Face and Gesture Recognition*, 1994, pp. 296–301.

[43] V. Chandrasekhar, D. M. Chen, A. Lin, G. Takacs, S. S. Tsai, N. M. Cheung, Y. Reznik, R. Grzeszczuk, and B. Girod, "Comparison of local feature descriptors for mobile visual search," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, Hong Kong, Sept. 2010.

[44] Y. Reznik, V. Chandrasekhar, G. Takacs, D. M. Chen, S. S. Tsai, R. Grzeszczuk, and B. Girod, "Fast quantization and matching of histogram-based image features," in *Proc. SPIE Workshop Applications of Digital Image Processing (ADIP)*, San Diego, CA, Aug. 2010.

[45] V. Chandrasekhar, G. Takacs, D. M. Chen, S. S. Tsai, R. Grzeszczuk, Y. Reznik, and B. Girod, "Compressed histogram of gradients: A low bitrate descriptor," *Int. J. Comput. Vis. (Special Issue on Mobile Vision)*, to be published.

[46] E. Tola, V. Lepetit, and P. Fua, "A fast local descriptor for dense matching," in *Proc. Conf. Computer Vision and Pattern Recognition*, 2008.

[47] S. S. Tsai, D. M. Chen, G. Takacs, V. Chandrasekhar, J. P. Singh, and B. Girod, "Location coding for mobile image retreival systems," in *Proc. Int. Mobile Multimedia Communications Conf. (MobiMedia)*, London, U.K., Sept. 2009.

[48] S. S. Tsai, D. Chen, G. Takacs, V. Chandrasekhar, J. P. Singh, and B. Girod, "Location coding for mobile image retrieval," in *Proc. Int. Mobile Multimedia Communications Conf. (Mobimedia)*, London, U.K., Sept. 2009.

[49] G. Schindler, M. Brown, and R. Szeliski, "City-scale location recognition," in *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR)*, Minneapolis, MN, June 2007.

[50] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman, "Total recall: Automatic query expansion with a generative feature model for object retrieval," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Minneapolis, MN, 2007.

[51] T. Yeh, J. J. Lee, and T. J. Darrell, "Adaptive vocabulary forests for dynamic indexing and category learning," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, Rio de Janeiro, Brazil, 2007.

[52] O. Chum, J. Philbin, and A. Zisserman, "Near duplicate image detection: min-hash and tf-idf weighting," in *Proc. British Machine Vision Conf. (BMVC)*, Leeds, U.K., Sept. 2008.

[53] O. Chum, M. Perdoch, and J. Matas, "Geometric min-hashing: Finding a (thick) needle in a haystack," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Miami, FL, June 2009.

[54] X. Zhang, Z. Li, L. Zhang, W.-Y. Ma, and H.-Y. Shum, "Efficient indexing for large-scale visual search," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, Kyoto, Japan, Sept. 2009.

[55] A. Mikulik, M. Perdoch, O. Chum, and J. Matas, "Learning a fine vocabulary," in *Proc. European Conf. Computer Vision (ECCV)*, Crete, Greece, Sept. 2010.

[56] P. Turcot and D. Lowe, "Better matching with fewer features: The selection of useful features in large database recognition problems," in *Proc. Workshop Emergent Issues in Large Amounts of Visual Data (WS-LAVD) and Int. Conf. Computer Vision (ICCV)*, Kyoto, Japan, Oct. 2009.

[57] Y. Li, N. Snavely, and D. P. Huttenlocher, "Location recognition using prioritized feature matching," in *Proc. European Conf. Computer Vision (ECCV)*, Crete, Greece, Sept. 2010.

[58] 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 *Proc. IEEE Data Compression Conf. (DCC)*, Snowbird, UT, Mar. 2010.

[59] H. Jegou, M. Douze, and C. Schmid, "Packing bag-of-features," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, Kyoto, Japan, Sept. 2009.

[60] H. Jegou, M. Douze, C. Schmid, and P. Perez, "Aggregating local descriptors into a compact image representation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, San Francisco, CA, June 2010.

[61] F. Perronnin, Y. Liu, J. Sanchez, and H. Poirier, "Large-scale image retrieval with compressed fisher vectors," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, San Francisco, CA, June 2010.

[62] J. Zobel and A. Moffat, "Inverted files for text search engines," *ACM Comput. Surveys*, vol. 38, no. 2, p. 6, July 2006.

[63] A. Said, "Comparative analysis of arithmetic coding computational complexity," HP Labs Tech. Rep., 2004.

[64] V. N. Anh and A. Moffat, "Inverted index compression using word-aligned binary codes," *Inform. Retrieval*, vol. 8, no. 1, pp. 151–166, Jan. 2005.

[65] A. Moffat and V. N. Anh, "Binary codes for non-uniform sources," in *Proc. IEEE Data Compression Conf. (DCC)*, Snowbird, UT, Mar. 2005.

[66] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[67] F. Schaffalitzky and A. Zisserman, "Multi-view matching for unordered image sets, or 'how do i organize my holiday snaps?'," in *Proc. 7th European Conf. Computer Vision-Part I (ECCV '02)*, London, U.K. Springer-Verlag, 2002, pp. 414–431.

[68] V. Ferrari, T. Tuytelaars, and L. Van Gool, "Simultaneous object recognition and segmentation by image exploration," in *Proc. European Conf. Computer Vision (ECCV)*, Prague, Czech Republic, May 2004.

[69] Z. Wu, Q. Ke, M. Isard, and J. Sun, "Bundling features for large scale partial-duplicate web image search," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Miami, FL, 2009.

[70] O. Chum, J. Matas, and J. V. Kittler, "Locally optimized RANSAC," in *Proc. DAGM Symp.*, Magdeburg, Germany, Sept. 2003.

[71] O. Chum, T. Werner, and J. Matas, "Epipolar geometry estimation via ransac benefits from the oriented epipolar constraint," in *Proc. Int. Conf. Pattern Recognition (ICPR)*, Cambridge, U.K., Aug. 2004.

[72] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Minneapolis, MN, 2007.

[73] 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 *Proc. IEEE Int. Conf. Image Processing (ICIP)*, Hong Kong, Sept. 2010.

[74] F. Schaffalitzky and A. Zisserman, "Automated scene matching in movies," in *Proc. ACM Int. Conf. Image and Video Retrieval*, London, U.K., July 2002.

[75] D. M. Chen, S. S. Tsai, R. Vedantham, R. Grzeszczuk, and B. Girod. (2008, Apr.). CD cover database—Query images [Online]. Available: http://vcui2.nokiapaloalto.com/ dchen/cibr/testimages/

[76] D. M. Chen, S. S. Tsai, R. Grzeszczuk, R. Vedantham, and B. Girod, "Streaming mobile augmented reality on mobile phones," in *Proc. Int. Symp. Mixed and Augmented Reality (ISMAR)*, Orlando, FL, Oct. 2009. [SP]