Memory-Efficient Image Databases for Mobile Visual Search

David M. Chen
Stanford University

Bernd Girod
Stanford University

Storing a memory-efficient database of image signatures entirely on a mobile device can enable fast local queries regardless of external conditions, such as a slow network or congested server.

Mobile visual search (MVS) systems recognize objects in the user's local environment, retrieve interesting and important information about the objects, and overlay the information in the mobile device's viewfinder. Figure 1 shows a typical example of an MVS system. The system recognizes outdoor buildings, overlays the address and phone number of each building, and shows the building's location on a map of the local neighborhood. MVS systems have also been developed for recognizing and augmenting media covers, product packages, billboards, artwork, and clothing, among other categories of objects. Recent commercial deployments of MVS technologies include Amazon Flow, Kooaba Visual Search, Google Goggles, Nokia Point and Find, and Layar Browser.

For accurate object recognition in MVS, images captured by the device's camera are compared against a database of labeled images. A near-real-time response is desired to provide seamless and continuous augmentation. Existing MVS systems typically query a database hosted on a remote server and can achieve a low latency, around 1 second, when the network connection is fast and when the server is highly responsive. However, slow transmissions over a wireless network or congestion on a busy server can severely degrade the user experience.

To address this problem, we explain how a memory-efficient database of image signatures stored entirely on a mobile device can enable fast local queries. A locally stored database can provide fast recognition anywhere and anytime, regardless of conditions outside the mobile device. To realize this goal, the image signatures stored in the local database must be extremely compact to fit in the small amount of memory available on the mobile device, capable of efficient comparisons across a large database, and highly discriminative to provide robust recognition for challenging queries. With compact image signatures, a mobile device can store a database containing images of outdoor landmarks, book covers, or product packages, among many more practical examples. When the database requires an update in response to changes in the user's environment or interests, the same signatures should support incremental database updates. Ideally, when server and network conditions improve, these compact signatures can be transmitted to a remote server for expanded queries against a remote database.

In this article, we present four methods recently developed for constructing a compact database from local image-based features and compare their retrieval performances: tree histogram coding (THC),[1] inverted index coding (IIC),[2] residual enhanced visual vector (REVV),[3] and scalable compressed fisher vector (SCFV).[4] Both THC and IIC use compression techniques in conjunction with a bag-of-visual-words histogram to generate compact and discriminative global image signatures. These two methods require the storage of a codebook in the mobile device's memory and decoding of compressed signatures during a query. In contrast, compact REVV and SCFV signatures are generated from bag-of-visual-words residuals. While achieving the same high-level retrieval performance as THC and IIC, REVV and SCFV utilize a much smaller codebook and perform comparisons directly in the compressed domain. We also present several new improvements to REVV and utilize them in an on-device image retrieval system for the Android platform that achieves low recognition latencies and is parsimonious

1070-986X/14/$31.00 © 2014 IEEE

in its memory usage. An on-device retrieval framework becomes increasingly attractive as processor speeds and memory capacity of available mobile devices rapidly improve.

On-Device Image Retrieval

Figure 2 shows an on-device image retrieval and augmentation pipeline designed for low-latency MVS. First, the motion of objects between viewfinder frames is analyzed to detect periods of low motion, when the user is most likely to be interested in the viewfinder's contents. In each identified low-motion interval, the feature-rich keyframes are automatically selected by maximizing the expected number of local features. Then, local features such as scale-invariant feature transform (SIFT)[5] or speeded-up robust features (SURF)[6] are extracted from the selected frames.

The next few steps in the pipeline shown in Figure 2 cover the image retrieval process using the local image features. Comparing the local features directly would be too computationally expensive and slow, so a global signature is usually generated from many local features extracted for each image. A well-designed global signature can summarize the most important statistics of the local features and has a compact representation of the overall image characteristics to enable fast comparisons. By comparing the query global signature against all database global signatures, a ranked list of database candidates is produced. The candidates at the top of this ranked list are further tested for geometric consistency of local feature matches with respect to the query image.

Lastly, relevant augmentations are drawn for the recognized objects in the viewfinder based on the retrieved information. The positions of the recognized objects are continuously updated using the same motion analysis algorithm that was previously used to detect low-motion intervals.

Crucial to the success of the entire pipeline is a global image signature that should be efficient to compare, highly discriminative for accurate differentiation of images from one another in a large database, and compact for convenient storage and fast access in memory. Compared with a commodity server that often has 16 to 64 Gbytes of RAM, a mobile device generally has one to two orders of magnitude less RAM. The MVS application must share this small amount of RAM with many other applications concurrently running on the device.

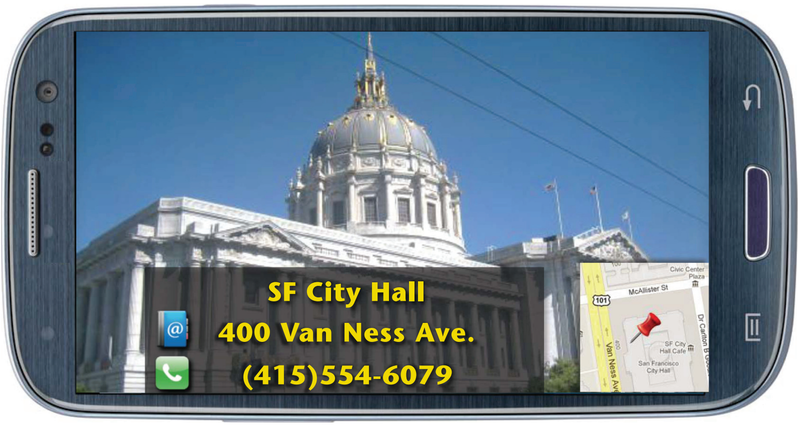


Figure 1. Mobile visual search (MVS) system. The system recognizes outdoor buildings and augments the viewfinder with useful information about each recognized building.

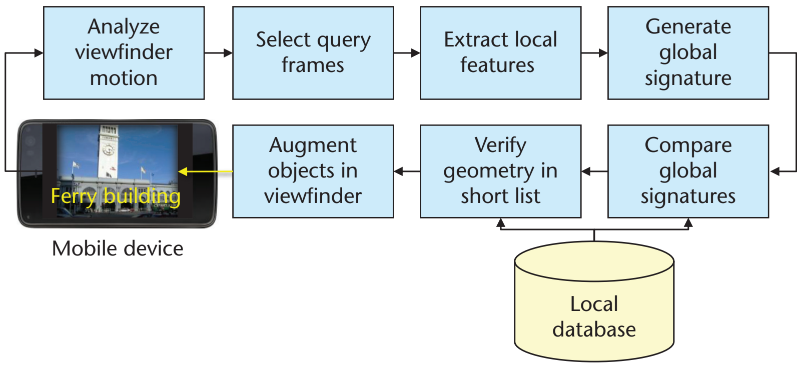

Analyze viewfinder motion
Select query frames
Extract local features
Generate global signature
Ferry building
Mobile device
Augment objects in viewfinder
Verify geometry in short list
Compare global signatures
Local database

Figure 2. On-device recognition and augmentation pipeline for a mobile visual search system. Crucial to the success of the entire pipeline is a global image signature, which is generated from a set of local features extracted for each image.

Given such tight constraints on RAM capacity, generating a compact visual database is critically important. A compact database representation will provide the following major capabilities:

- More images can be indexed in the database, increasing the image search range.
- Multiple databases can be stored in RAM, allowing MVS systems to search for multiple classes of objects.
- Each database can be quickly loaded into RAM from the storage card.
- A memory-efficient database is faster to search because fewer memory accesses are required to complete each query.

Each of the four methods we discuss here attempts to build compact visual databases.

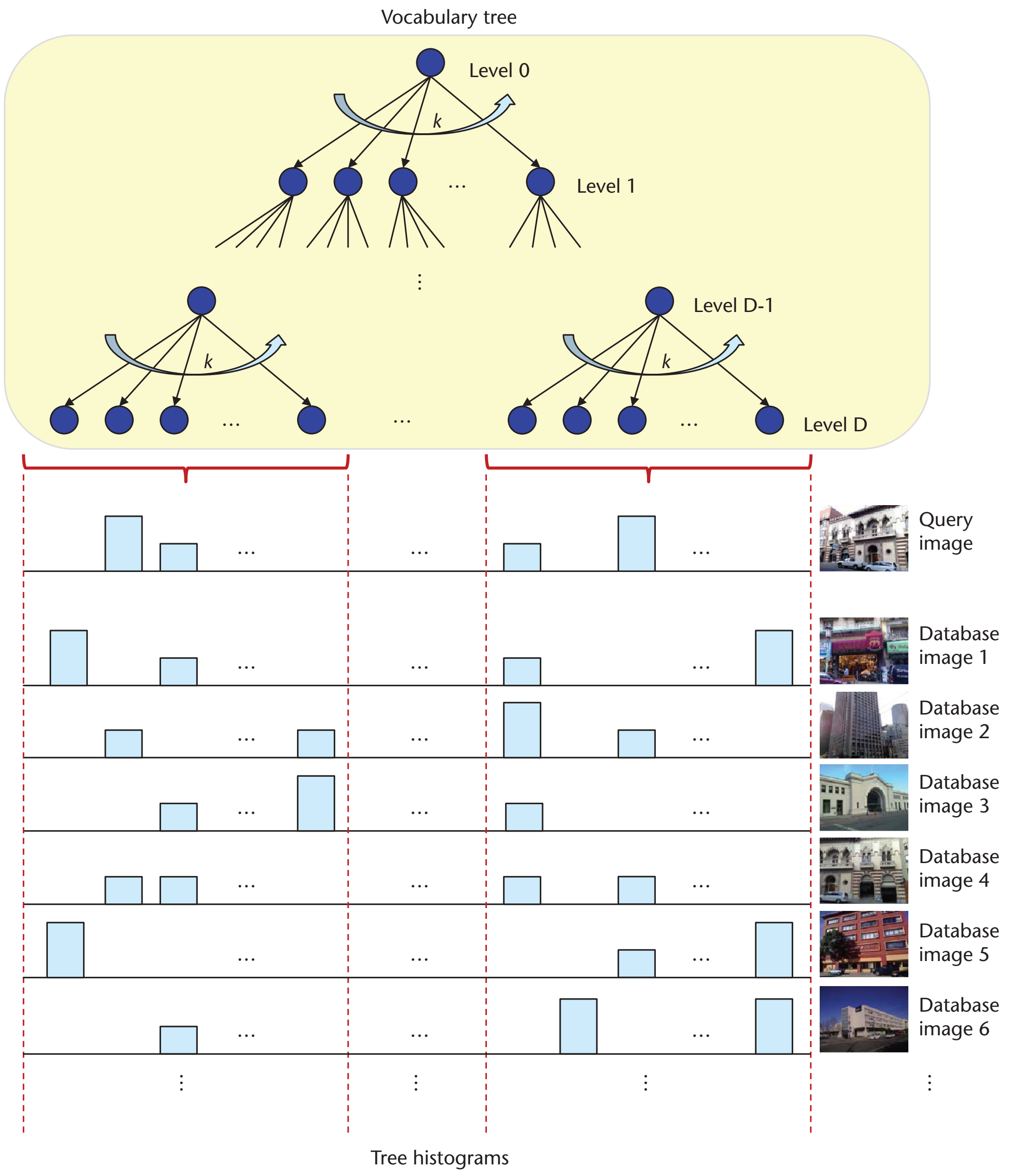


Figure 3. Vocabulary tree and the associated tree histograms for a set of images. A vocabulary tree with depth D *and branch factor* k *has* k^D *leaf nodes at the bottom level* D.

Bag-of-Visual-Words Histograms

Many popular image retrieval systems are based on bag-of-visual-words histograms.[7–10] A codebook of visual words is generated from a sample set of local feature descriptors. For good retrieval performance, the codebook must contain a large number of visual words, for example, from 100,000 to 1 million. To create such a large codebook, hierarchical k-means clustering can be employed,[11] and the output of the hierarchical divisive clustering process is a tree-structured vector quantizer commonly called a *vocabulary tree*.[8]

A vocabulary tree with depth D and a branch factor k has k^D leaf nodes at the bottom level D, as depicted in Figure 3. After quantizing all feature descriptors for an image to the nearest leaf nodes using a greedy search,[9] the resulting tree histogram acts as a global image signature. For the tree parameters $k = 10$ and $D = 6$, there are

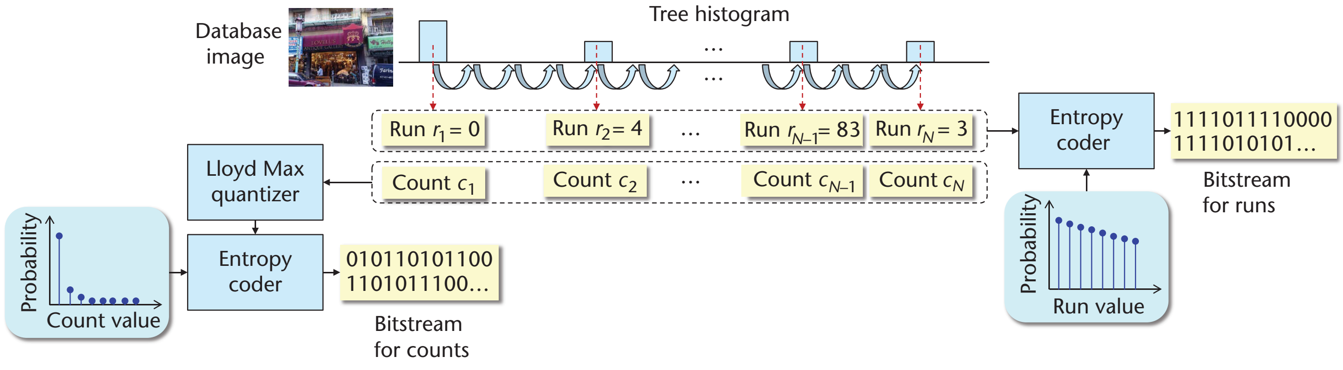


Figure 4. Encoding the tree histogram for an image with tree histogram coding (THC).

1 million leaf nodes. Because there are typically several hundred local features per image, a tree histogram for the 1 million leaf nodes is sparsely filled. This type of sparsity leads to both excellent retrieval performance and efficient calculations of histogram intersection scores.[8] During histogram intersection, the count or term frequency (TF) in each histogram bin can be multiplied by an inverse document frequency (IDF) weight, resulting in the popular TF-IDF scoring method.[12]

Storing a vocabulary tree with 1 million leaf nodes requires approximately 64 to 128 Mbytes of RAM on a mobile device, depending on the dimensionality of some popular feature descriptors.[1] Although storing a tree of this size is feasible with most of today's mobile devices, it will greatly reduce the memory available for storing tree histograms and running other applications concurrently on the device. A large tree also hinders the quick launch of MVS applications because it may take tens of seconds to load a large tree from the storage card into RAM. Thus, it is desirable to achieve the same high level of retrieval performance as a vocabulary tree but require a substantially smaller codebook. We will revisit this issue when discussing bag-of-visual-words residuals later on.

Storing the tree histograms efficiently in memory is possible with tree histogram coding (THC)[1] or inverted index coding (IIC).[2] THC takes a direct approach to compressing the tree histograms, and IIC first transforms the tree histograms into an inverted index structure and then compresses the index. Both methods require some selective decompression during a query. Direct comparisons in the compressed domain are desirable for faster retrieval, which we discuss in more detail in the "Bag-of-Visual-Words Residuals" section.

Tree Histogram Coding

Because tree histograms are sparse, a memory-efficient representation of the tree histograms should directly exploit this sparsity of the data. Assume that the leaf nodes of the vocabulary tree are enumerated in depth-first order. Then, THC encodes the runs between consecutive nonempty histogram bins and the counts within those bins (see Figure 4).[1] The sequence of run values is compressed with an entropy coder using the probability distribution of runs. The sequence of count values is first quantized with a Lloyd-Max quantizer and then compressed with an entropy coder based on the probability distribution of quantized counts. Compressed bitstreams for the runs and the counts of the database tree histograms can then be efficiently stored in RAM. During a query, the compressed bitstreams are decoded to enable comparisons against the tree histogram for the current query image.

There is a design trade-off between achieving the largest compression gain and reducing the decoding time during a query, which greatly impacts the overall system performance. On the one hand, an arithmetic coder (AC) can be used to encode the runs and quantized counts to levels that are close to the respective information theoretic entropy limits. On the other hand, entropy coders that compress data into byte-aligned codewords[13] offer much faster decoding than AC, at the cost of slightly lower coding efficiency. For low-latency MVS, it is usually preferable to use a byte-aligned entropy coder to achieve low decoding delays while still maintaining a high compression ratio.

Inverted Index Coding

An inverted index enables fast computation of the similarity scores between the query and

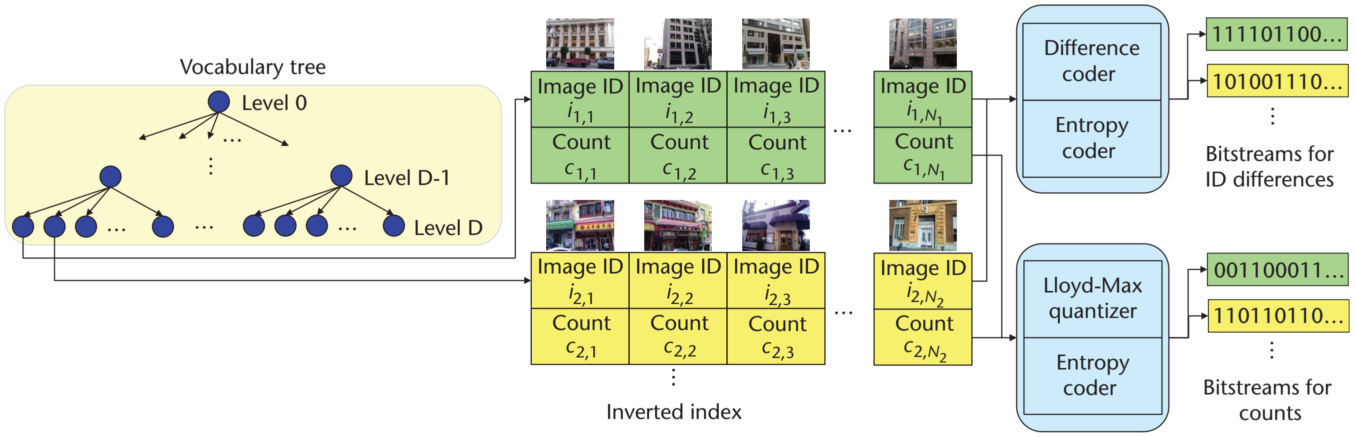


Figure 5. Encoding the inverted index associated with a vocabulary tree with inverted index coding (IIC).

database tree histograms. Figure 5 shows the inverted lists for two leaf nodes in a vocabulary tree. For each leaf node, the associated inverted list contains the identifiers (IDs) of all images with at least one feature descriptor visiting the leaf node and the counts of how often the images have visited. Because only a small fraction of the database images visit each leaf node, the length of an inverted list is much smaller than the size of the database, which is a key property that enables fast score computation.

A memory-efficient representation of the index is achievable using IIC.[2] Assume the image IDs are sorted in increasing order—for example, $i_{1,1} < i_{1,2} < \ldots < i_{1,N_1}$ for IDs in the first list. Then, it is sufficient to encode the differences between consecutive IDs with an entropy coder. Difference coding is similar in spirit to the coding of runs used in THC, although the underlying statistics are different. The counts can be effectively compressed using a Lloyd-Max quantizer and an entropy coder. Image IDs and counts for each leaf node are encoded independently so that only the compressed lists for leaf nodes visited during a query need to be decompressed. As in THC, there is a trade-off between high compression efficiency and faster decoding. Byte-aligned coders can compress the image database almost as well as an arithmetic coder, while resulting in much shorter decoding delays.[2]

Bag-of-Visual-Words Residuals

In contrast to the bag-of-visual-words histograms discussed previously, a new trend in image retrieval is generating compact and discriminative bag-of-visual-words residuals.[3,4,14,15] The residual representation also uses a codebook of visual words, but it is typically of a much smaller size (128 to 256 visual words). Such a small codebook reduces the memory requirement of the vector quantizer by a several orders of magnitude compared with bag-of-visual-words histograms. A histogram formed over such a small codebook, however, would not be discriminative enough for large-scale retrieval. Instead, the primary signals used to generate discriminative signatures are now the quantization errors or residuals between the feature descriptors and their nearest visual words. It is possible to generate compact and discriminative residuals with REVV[3] or SCFV.[4] In contrast to THC and IIC, REVV and SCFV enable signature comparisons directly in the compressed domain.

Residual Enhanced Visual Vector

The algorithm for generating and comparing REVV signatures is illustrated in Figure 6. First, using a codebook of k visual words—for example, $k = 128$ to $k = 256$—an image's local feature descriptors are quantized to the nearest visual words, and residual vectors between the descriptors and their nearest visual words are computed. Second, for each visual word, the mean of the residual vectors for that visual word is calculated. Third, a nonlinear power law transform is applied to the residual vectors to reduce the detrimental influence of peaky components. Fourth, for each visual word, the residual vector is projected onto a set of eigenvectors obtained by linear discriminant analysis (LDA). Finally, the REVV vectors are binarized based on the sign of each component, and

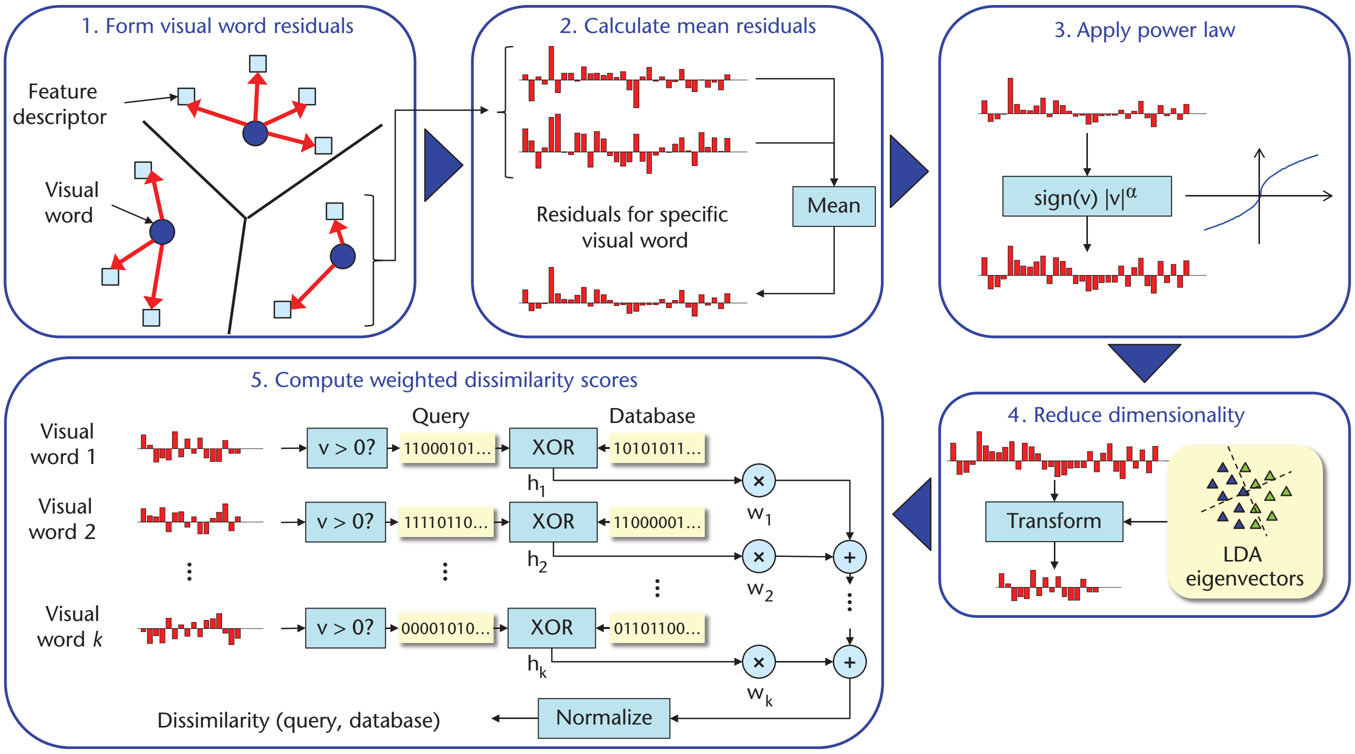


Figure 6. Pipeline for generating and comparing residual enhanced visual vector (REVV) signatures.

weighted correlations are calculated for the query REVV signature and every database REVV signature.

To improve retrieval performance, the Hamming distance h_i at the ith visual word is weighted by a factor w_i that distinguishes between informative and noninformative observations. Specifically, we use $w_i = w_i^1 \cdot w_i^2$, where w_i^1 and w_i^2 are defined as follows. First, $w_i^1 = [P_m(h_i)/(P_m(h_i) + P_{nm}(h_i))]^\beta$, where h_i is the Hamming distance between the binarized query and database residuals for the ith visual word, and $P_m(h)$ and $P_{nm}(h)$ are the probabilities that matching and nonmatching residuals have Hamming distance h, respectively. For $\beta > 1$, this power law transformation rewards visual words that generate a low Hamming distance. Second, $w_i^2 = \log(m \cdot N_{\max}/(N(q_{i,1}) + \ldots + N(q_{i,m})))$. Here, $q_{i,j}$ refers to the jth part of the binarized query residual for the ith visual word. $N(q_{i,j})$ $(j = 1, \ldots, m)$ is the number of times that the jth part of a binarized database residual lies within Hamming distance h_t of $q_{i,j}$ at the ith visual word. $N_{\max}$ is a large number that serves as an upper bound for $N(q_{i,j})$—for example, $N_{\max}$ is the number of images in the database. This weight can be interpreted as an IDF weight for patterns in Hamming space.

When all the weighted Hamming distances are added, a normalization is required to account for different number of features in different database images. In previous work,[7] we normalized the score for each database image by $N_d = \sqrt{d_{\text{LDA}}|I_d|}$, where d_{LDA} is the number of LDA eigenvectors per visual word and I_d is the set of visual words visited by a database image. Because the value $|I_d|$ can vary significantly between database images, we apply a variance-stabilizing transformation[16] to generate the new normalization factor: $N_d = (d_{\text{LDA}}|I_d|)^\gamma$, where $0 < \gamma < 0.5$.

Compressed Scalable Fisher Vector

Recently introduced, the SCFV is also a global signature based on visual word residuals. Because of its excellent retrieval performance and compactness, the SCFV has been selected by the MPEG Compact Descriptors for Visual Search (CDVS) subgroup for adoption into the CDVS Test Model.[17] SCFV signatures are generated by the pipeline depicted in Figure 7, which has similarities to the pipeline for generating

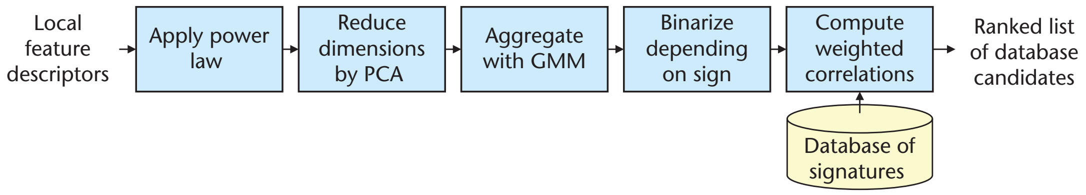

Local feature descriptors
Apply power law
Reduce dimensions by PCA
Aggregate with GMM
Binarize depending on sign
Compute weighted correlations
Ranked list of database candidates
Database of signatures

Figure 7. Pipeline for generating and comparing scalable compressed fisher vector (SCFV) signatures.

REVV signatures. First, the local feature descriptors are transformed by a power law that reduces the detrimental influence of peaky components. Second, each descriptor vector is reduced in dimensionality using principal component analysis (PCA). Next, the PCA-projected descriptors are quantized using a Gaussian mixture model (GMM). Gradient vectors with respect to the GMM centroids are generated for the subsequent formation of a compact and discriminative signature. Dimensionality reduction of the feature descriptor is performed prior to the GMM stage because the GMM parameters can be more effectively learned on vectors of lower dimensionality. Then, the residual vectors are binarized depending on the sign of each component. Finally, weighted correlations between the query signature and database signatures are computed to generate a ranked list of database candidates.

SCFV offers a scalable representation that can be adjusted depending on the memory budget. For a small memory capacity, SCFV selects a small number of centroids in the GMM and retains only the gradient vectors for these centroids. As memory capacity increases, the set of selected centroids can be enlarged to yield a more discriminative signature. The basic version of the SCFV utilizes the gradient vector with respect to the Gaussian mean in the GMM. If additional memory is available, the discriminative capability of the SCFV can be enhanced by also utilizing the gradient vector with respect to the Gaussian variance.

Performance Evaluation of Compact Databases

To assess the performance of an MVS system that uses on-device image retrieval, we first obtained the retrieval performance and memory usage on two publicly available datasets that are commonly used for benchmarking. Then, we measured the query latency in an Android implementation of on-device image retrieval.

Evaluation for Stanford Mobile Visual Search Dataset

We measured the retrieval performance on the publicly available Stanford Mobile Visual Search (SMVS) dataset,[18] which contains 3,300 query images and a database of 1 million images. Figures 8a and 8b plot the retrieval recall and precision versus database size for THC, IIC, and REVV on the SMVS dataset. Both THC and IIC use a vocabulary tree with 1 million leaf nodes, multipath greedy-10 nearest neighbor search,[9] soft binning with three leaf nodes per feature descriptor,[10] and TF-IDF weighting.[12] These parameter settings for the vocabulary tree have been found to yield high retrieval performance. REVV uses a codebook of 128 visual words. All three schemes use SURF features with 64-dimensional descriptors.[6] As we can see from Figure 8, REVV achieves comparable retrieval recall and precision as THC and IIC with a much smaller codebook.

Figure 8c plots the RAM usage for a database of 10,000 images contained in the SMVS dataset and clearly shows the advantage of REVV over THC and IIC for on-device image retrieval. Memory usage is divided between the database signatures and the auxiliary data. For THC and IIC, the auxiliary data consists of a vocabulary tree and a set of IDF weights. For REVV, the auxiliary data consists of a codebook of visual words, a set of LDA eigenvectors, and weights used for the similarity calculation. Both THC and IIC use substantially more RAM than REVV in the database signatures and auxiliary data. Thus, it is much easier to deploy a REVV-based image retrieval pipeline on a mobile device with small RAM capacity.

Evaluation for MPEG Compact Descriptors for Visual Search

We also compared the retrieval performance of THC, IIC, REVV, and SCFV on the MPEG CDVS dataset,[17] which contains 8,313 query images from five classes (graphics, paintings, video frames, landmarks, and common objects) and a

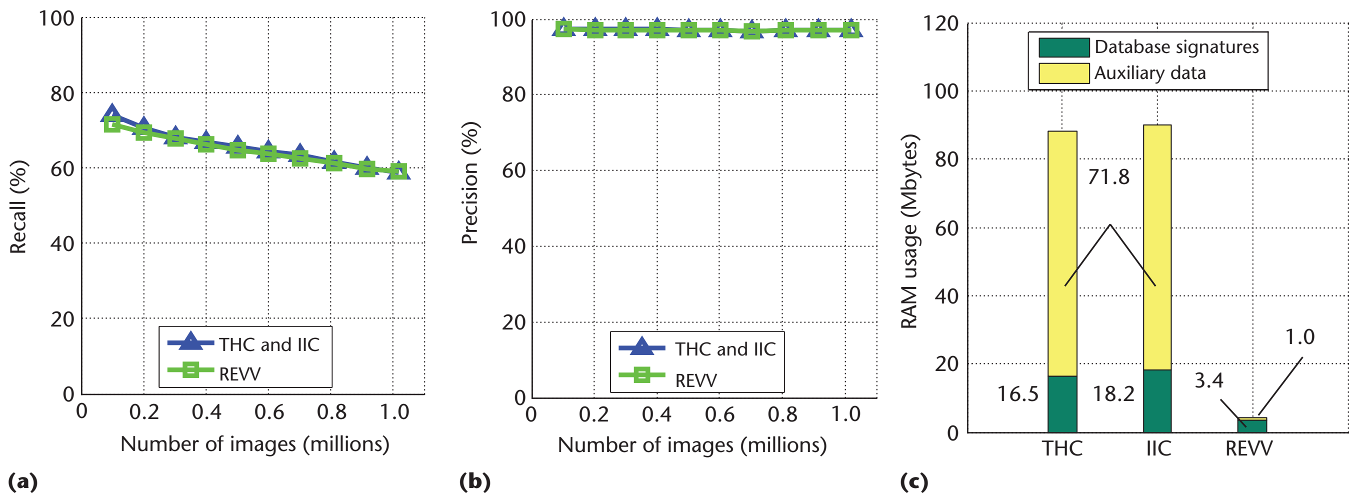


Figure 8. Comparison of THC, IIC, and REVV on the Stanford Mobile Visual Search (SMVS) dataset with SURF features in terms of (a) retrieval recall, (b) retrieval precision, and (c) RAM usage for a database of 10,000 images sampled from the dataset.

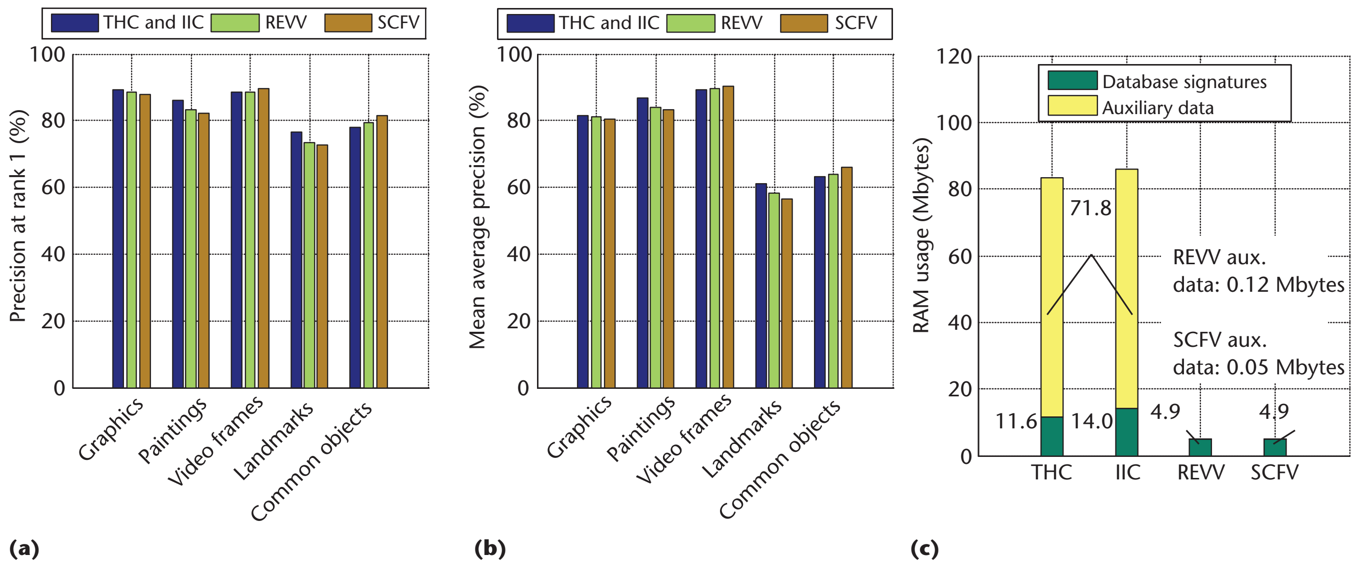
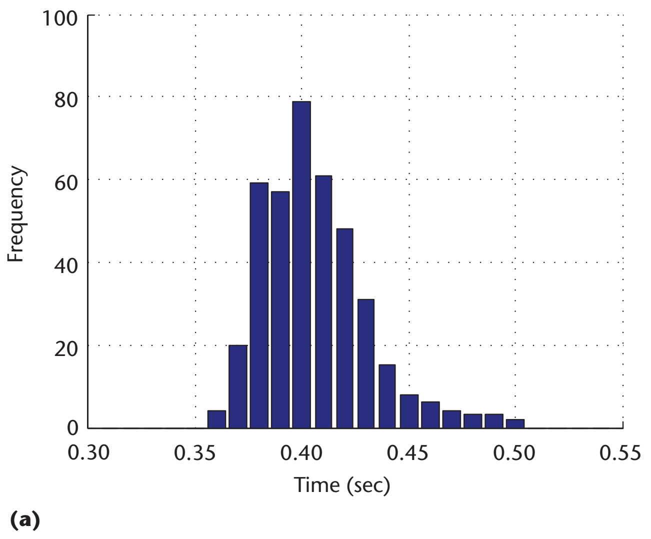


Figure 9. Comparison of THC, IIC, REVV, and SCFV on MPEG CDVS dataset with SIFT features in terms of (a) precision at rank 1, (b) mean average precision, and (c) RAM usage for a database of 10,000 images sampled from the dataset.

database of 1 million images. In this experiment, all four schemes use the same SIFT features.[5] Figures 9 shows the precision at rank 1 and the mean average precision of THC, IIC, REVV, and SCFV for the five different classes. Overall, all four schemes obtain comparable retrieval performance. Figure 9c plots the RAM usage for a database of 10,000 images contained in the MPEG CDVS dataset. Similar to the results of the SMVS dataset, both THC and IIC use substantially more RAM than REVV and SCFV in the database signatures and auxiliary data.

Implementation on Android Smartphone

Using REVV, we have implemented an MVS system for recognizing outdoor landmarks and media covers on a Samsung Galaxy S3 smartphone, which has a 1.4 GHz processor and 1 Gbyte of RAM. A database of 10,000 images was indexed on the device. We measured the system latency for 400 different queries of media covers and landmarks, and we plotted the distribution of latencies in Figure 10a. Variations in the latency are mainly caused by different numbers of local features for the different objects. On average, each query required



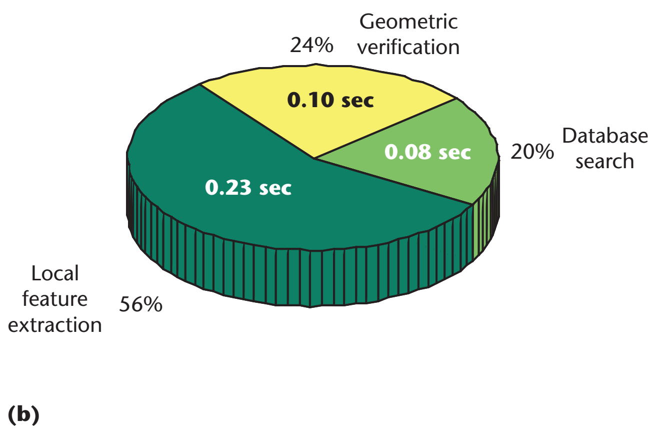


Figure 10. Android smartphone implementation. (a) Distribution of recognition latencies for 400 different queries. (b) Percentage of time spent on feature extraction, database search, and geometric verification.

0.4 seconds, with 56, 20, and 24 percent of the time spent on feature extraction, database search, and geometric verification, respectively (see Figure 10b). Importantly, this low latency can be achieved anywhere and anytime, independent of any adverse network conditions or server congestion.

Conclusions and Outlook

Our experimental results show that low-latency MVS systems can be constructed with local features and memory-efficient image databases. On-device image retrieval enables object recognition with low processing delay and without any potentially adverse effects from poor network conditions or server congestion. In particular, we have shown that compact and discriminative bag-of-visual-words residuals can be created with a small codebook, enabling an effective, easy-to-deploy image retrieval pipeline.

This work has focused on retrieval using a database stored locally on the device, without the need to query a remote server. Further study is required on how to efficiently update the local database on the mobile device with new data generated and stored on a remote server, including how to perform the updates during periods when the network conditions are favorable. Similarly, efficient on-device caching, including how to quickly transfer a compact database from the storage card to RAM in response to changes in the user location or other environmental context, deserves more attention. Compact global signatures can be beneficial for fast database updates and caching because small amounts of data need to be transmitted, transferred, and stored. **MM**

Acknowledgments

We thank the editors and reviewers for their insightful comments and suggestions, which greatly helped to improve this article.

References

1. D. Chen et al., "Tree Histogram Coding for Mobile Image Matching," *Proc. IEEE Data Compression Conf.,* IEEE CS, 2009, pp. 143–152.
2. D. Chen et al., "Inverted Index Compression for Scalable Image Matching," *Proc. IEEE Data Compression Conf.,* IEEE CS, 2010, p. 525.
3. D. Chen et al., "Residual Enhanced Visual Vector as a Compact Signature for Mobile Visual Search," *Signal Processing,* vol. 93, no. 8, 2012, pp. 2316–2327.
4. J. Lin et al., "Peking University response to CE1: Performance improvements of the Scalable Compressed Fisher Vector," ISO/IEC JTCI/SC29/WG11 MPEG2013/M28061, Jan. 2013.
5. D.G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int'l J. Computer Vision,* vol. 60, no. 2, 2004, pp. 91–110.
6. H. Bay et al., "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding,* vol. 110, no. 3, 2008, pp. 346–359.

7. J. Sivic and A. Zisserman, "Video Google: A Text Retrieval Approach to Object Matching in Videos," *Proc. 9th IEEE Conf. Computer Vision,* vol. 2, IEEE CS, 2003, p. 1470.
8. D. Nister and H. Stewenius, "Scalable Recognition with a Vocabulary Tree," *Proc. IEEE Conf. Computer Vision and Pattern Recognition* (CVPR), IEEE CS, 2006, pp. 2161–2168.
9. G. Schindler, M. Brown, and R. Szeliski, "City-Scale Location Recognition," *Proc. IEEE Conf. Computer Vision and Pattern Recognition* (CVPR), IEEE CS, 2007, pp. 1–7.
10. J. Philbin et al., "Lost in Quantization: Improving Particular Object Retrieval in Large Scale Image Databases," *Proc. IEEE Conf. Computer Vision and Pattern Recognition* (CVPR), IEEE CS, 2008, pp. 1–8.
11. T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning,* Springer, 2009.
12. G. Salton and C. Buckley, "Term-Weighting Approaches in Automatic Text Retrieval," *Information Processing and Management,* vol. 24, no. 5, 1988, pp. 513–523.
13. V.N. Anh and A. Moffat, "Inverted Index Compression Using Word-Aligned Binary Codes," *Information Retrieval,* vol. 8, no. 1, 2005, pp. 151–166.
14. H. Jegou et al., "Aggregating Local Descriptors into a Compact Image Representation," *Proc. IEEE Conf. Computer Vision and Pattern Recognition* (CVPR), IEEE CS, 2010, pp. 3304–3311.
15. F. Perronnin et al., "Large-Scale Image Retrieval with Compressed Fisher Vectors," *Proc. IEEE Conf. Computer Vision and Pattern Recognition* (CVPR), IEEE CS, 2010, pp. 3384–3391.
16. F.J. Anscombe, "The Transformation of Poisson, Binomial and Negative Binomial Data," *Biometrika,* vol. 35, Mar. 1948, pp. 246–254.
17. Y. Reznik, G. Cordara, and M. Bober, "Evaluation framework for Compact Descriptors for Visual Search," ISO/IEC JTCI/SC29/WG11 N12202, July 2011.
18. V. Chandrasekhar et al., "The Stanford Mobile Visual Search Data Set," *Proc. ACM Conf. Multimedia Systems,* ACM, 2011, pp. 117–122.

David Chen is a doctoral student in electrical engineering at Stanford University. His research interests include computer vision, machine learning, and signal compression. He is currently a fellow of the Brown Institute for Media Innovation and a member of IEEE. Chen has an MS in Electrical Engineering with an emphasis on image and signal processing from Stanford. Contact him at dmchen@stanford.edu.

Bernd Girod is a professor of electrical engineering and (by courtesy) computer science at Stanford University. He also serves Stanford's School of Engineering as senior associate dean for online learning, is director of the Brown Institute for Media Innovation, and has been involved in several startup ventures. His research interests include networked media systems. Girod has a PhD in electrical engineering from the University of Hannover, Germany. He is a fellow of IEEE. Contact him at bgirod@stanford.edu.