

Survey of SIFT Compression Schemes

Vijay Chandrasekhar*, Mina Makar*, Gabriel Takacs*, David Chen*,
Sam S. Tsai*, Ngai-Man Cheung*, Radek Grzeszczuk†,
Yuriy Reznik‡, and Bernd Girod*

*Stanford University, † Nokia Research Center, CA ‡ Qualcomm Inc., CA

Abstract. Transmission and storage of local feature descriptors are of critical importance for mobile visual search applications. We perform a comprehensive survey of Scale Invariant Feature Transform (SIFT) compression schemes proposed in the literature and evaluate them in a common framework. Further, we compare the different schemes to the recently proposed low bit-rate Compressed Histogram of Gradients (CHoG) descriptor. We show that CHoG outperforms all SIFT compression schemes. We implement CHoG in a large-scale mobile image retrieval system and show that transmitting CHoG feature data are an order of magnitude smaller than transmitting SIFT descriptors or JPEG images.

1 Introduction

Local image features have become pervasive in the areas of computer vision and image retrieval. Feature compression is vital for reduction in storage, latency and transmission in mobile visual search applications.

Transmission time: For mobile visual search applications, bandwidth is a limiting factor. One approach used in mobile visual search applications is to transmit the JPEG compressed query image over the network, but this might be prohibitively expensive at low uplink speeds. An alternate approach is to extract feature descriptors on the phone, compress the descriptors and transmit them over the network. In [1], we show that such an approach can reduce the application latency by an order of magnitude.

Server-client Caching: For several applications, a subset of descriptors are stored in RAM for fast access. Takacs *et al.* [2] cache a set of descriptors on the mobile client to enable an outdoors mobile augmented reality experience. Having compact descriptors allows storage of a larger set of descriptors in main memory.

Server-side Storage: Image and video retrieval applications need query images to be matched against databases of billions of features. E.g., 1000 hours of video produces \sim 10 billion local descriptors. Storing 10 billion uncompressed SIFT [3] descriptors would require \sim 10 TB of storage. More compact descriptors will lead to faster file accesses.

SIFT is the most popular descriptor in computer vision for retrieval applications. In this work, we perform a comprehensive survey of SIFT compression schemes and evaluate them in a common framework.

1.1 Prior Work and Outline

We broadly classify SIFT compression schemes into three categories: Hashing, Transform Coding, and Vector Quantization.

Hashing: Locality Sensitive Hashing (LSH) [4, 5] is the most popular hashing technique for high dimensional descriptors. In [6], Torralba *et al.* build binary codes for high dimensional descriptors using machine learning techniques like Restricted Boltzmann Machines (RBM) and Similarity Sensitive Coding (SSC). In [7], Weiss *et al.* propose a scheme called Spectral Hashing (SH) that outperforms RBM and SSC based approaches. For each of these schemes, exact Euclidean distances are approximated or estimated by Hamming distances over binary codewords.

Transform Coding: In [8], we have studied dimensionality reduction of SIFT descriptors using the Karhunen-Loeve Transform (KLT) followed by entropy coding. The KLT-based coding is known to work best for data with Gaussian statistics. Since SIFT statistics are highly non-Gaussian, we also study the performance of a transform coding scheme based on Independent Component Analysis (ICA).

Vector Quantization: Vector Quantization (VQ) of SIFT features is most commonly used in the “bag-of-features” image retrieval framework. For such applications, SIFT features are quantized using flat k -means (FKM) or hierarchical k -means (HKM) [9] to form a bag of “visual words”. HKM or FKM can also be used for compression of descriptors for storage or transmission. Jegou *et al.* [10] in their recent work propose a scheme called Product Quantization (PQ), where the SIFT descriptor is divided into smaller blocks and VQ is performed on each block. Some hybrid schemes have also been proposed in the literature. In [11], Jegou *et al.* propose a scheme called Hamming Embedding (HE), where SIFT descriptors are first coarsely quantized using HKM, and binary hashes are used for refinement in each quantization cell.

In our own work [12, 1], we propose a framework for computing low bit-rate feature descriptors called CHoG. Gradient histograms are quantized using Huffman trees, Type Quantization or Lloyd Max VQ and compressed efficiently using fixed or variable length codes.

Here, we will evaluate the different SIFT compression schemes in a common framework. In Section 2, we survey the different SIFT compression schemes proposed in the literature. In Section 3, we review the CHoG descriptor. Finally, in Section 4, we evaluate the performance of the different schemes.

2 SIFT Compression Schemes

2.1 Hashing

Locality Sensitive Hashing: We use the LSH scheme proposed by Yeo *et al.* [4]. To build a hash, we first randomly generate a set of hyperplanes that pass through the origin. Each bit of the hash is then determined by which side of the hyperplane the SIFT descriptor lies. and the Hamming distance of their hashes.

Similarity Sensitive Coding: Torralba *et al.* [6] use machine learning techniques such as SSC and RBM to train binary codes for high dimensional descriptors. The Boosting SSC algorithm learns an embedding of the original Euclidean space into a binary space such that distances between vectors in the original

space are correlated with their Hamming distances in the binary space. Each bit of the hash is obtained as the output of a weak classifier.

Spectral Hashing: Weiss *et al.* propose Spectral Hashing in their recent work [7]. The spectral hashing scheme performs Principal Component Analysis (PCA) on the data and fits a multidimensional rectangle to it. The dimensions of the rectangle determine the hashing functions of each bit.

2.2 Transform Coding

Karhunen-Loeve Transform: Transform coding of SIFT descriptors was first proposed in [8]. The compression pipeline first applies a Karhunen-Loeve Transform (KLT) transform (or PCA) to decorrelate the different dimensions of the feature descriptor. Then, each dimension of the KLT vector is scalar quantized. The quantized coefficients of the descriptors are entropy coded with an arithmetic coder. In [8], it was observed that applying the KLT is effective at low rates, but hurts performance at high rates. At high rates, scalar quantization and entropy coding with no transform performs better.

The KLT gives a good rotation for scalar quantization for Gaussian statistics as the decorrelation aligns the Gaussian distribution with the symbol axes. The KLT is optimal for Gaussian data, causing the transformed coefficients to be statistically independent. However, the statistics for SIFT features are not Gaussian as shown in Figure 1. Our goal is to make the transformed coefficients as statistically independent as possible. Hence, we explore an ICA based transform which outperforms the KLT.

ICA based Transform: Under high-rate assumptions, Narozny *et al.* in [13] provide a framework to compute the optimal linear transform for making the transformed coefficients as independent as possible, without assuming Gaussian statistics of the input signal or orthogonality of the transform matrix. They define the Generalized Coding Gain (GCG) as the ratio between distortion-rate functions in case of no transform vs. applying the transform. The transform that maximizes the GCG also maximizes the Generalized Maximum Reducible Bits defined as

$$R_{\text{GMRB}} = R_{\mathbf{I}}(D) - R_{\mathbf{A}}(D) \\ = \frac{1}{d} I(X_1; \dots; X_d) - \frac{1}{d} I(Y_1; \dots; Y_d) - \frac{1}{2d} \log_2 \left(\frac{\det[\text{diag}(\mathbf{A}^{-T}\mathbf{A}^{-1})]}{\det[\mathbf{A}^{-T}\mathbf{A}^{-1}]} \right),$$

where $R_{\mathbf{I}}(D)$ and $R_{\mathbf{A}}(D)$ are rate-distortion functions in case of no transform and applying the transform respectively, d is the length of the descriptor, $\mathbf{X} = [X_1; \dots; X_d]$ is a vector which represents the signal to be encoded, $\mathbf{Y} = [Y_1; \dots; Y_d]$ represents the transform coefficients, $I(\cdot)$ is mutual information function and \mathbf{A} is the transform $d \times d$ matrix where $\mathbf{Y} = \mathbf{A}\mathbf{X}$. Thus, the optimal linear transform \mathbf{A}_{opt} is calculated as

$$\mathbf{A}_{\text{opt}} = \arg \min_{\mathbf{A}} I(Y_1; \dots; Y_d) + \frac{1}{2} \log_2 \left(\frac{\det[\text{diag}(\mathbf{A}^{-T}\mathbf{A}^{-1})]}{\det[\mathbf{A}^{-T}\mathbf{A}^{-1}]} \right)$$

The first term is non-negative and equals to zero if and only if the transform coefficients are independent. The matrix that minimizes this term is the solution to the ICA problem. Note that the second term is also non-negative and equal

to zero iff the columns of \mathbf{A}^{-1} are pairwise orthogonal. Hence, it is considered as a pseudo-distance to orthogonality of the transform matrix \mathbf{A} . We use the code provided by the authors in [13] to solve the minimization problem. For more details, the reader is referred to [13]. The transform is applied the same way as KLT, but we expect better compression efficiency due to the highly non-Gaussian statistics of the SIFT descriptors.

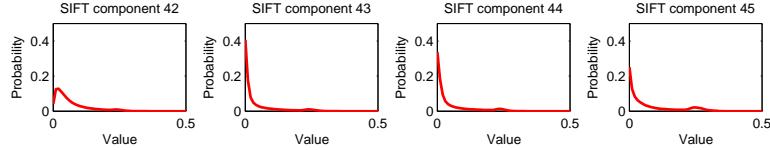


Fig. 1. We observe that the statistics of SIFT descriptors are non-Gaussian.

2.3 Vector Quantization

Product Quantization: Since the SIFT descriptor is high dimensional, Jegou *et al.* [10] propose a product quantizer which operates on lower dimensional subspaces and quantizes each subspace separately. The authors propose two variants of the scheme. The first scheme decomposes the SIFT descriptor directly into smaller blocks and performs VQ on each block. In the second scheme, the authors coarsely quantize the full descriptor with flat k -means or hierarchical k -means using 10^3 to 10^6 nodes. The residual is then quantized using a product quantizer. The two schemes perform comparably, and we consider the former scheme in our comparisons here.

In [10], the authors also investigate how different dimensions of the descriptor should be grouped together for good performance. The order that corresponds to grouping consecutive components together performs the best. Intuitively, this works well because histograms of consecutive cells are quantized together. There are two parameters used to control the bitrate: the number of blocks, denoted as B , and the size of the codebook for each block, denoted as C . Fixed length codes are used for each block and the bitrate of each descriptor is given by $B \times \lceil \log_2 C \rceil$. In Section 4, we consider $B = 1, 2, 4, 8, 16$ and $C = 16, 64, 256, 1024$. In the case of $B = 1$, the product quantizer reverts to flat k -means for the full descriptor.

Tree Structured Vector Quantization: Nistér and Stewénius [9] use HKM or a Tree Structured Vector Quantizer (TSVQ) to quantize SIFT descriptors and build a Inverted File System for fast indexing. Here, we use the same scheme for quantization and compression of SIFT descriptors. We quantize SIFT descriptors with a 10^6 node TSVQ with a branch factor (BF) of 10 and depth (D) of 6, requiring 20 bits per descriptor. A significantly larger TSVQ is not practical due to the size of the code book.

3 CHoG Descriptor

CHoG [12] is a Histogram of Gradients descriptor that is designed to work well at low bitrates. We highlight some key aspects of the descriptor here and readers are referred to [12, 1] for more details. First, we divide the patch into soft log polar spatial bins using DAISY configurations proposed in [14]. Next, the joint

(d_x, d_y) gradient histogram in each spatial bin is captured directly into the descriptor. CHoG histogram binning exploits the skew in gradient statistics that are observed for patches extracted around keypoints. Finally, CHoG retains the information in each spatial bin as a distribution. This allows the use of more effective distance measures like KL divergence, and more importantly, enables efficient quantization and compression. Typically, 9 to 13 spatial bins and 3 to 9 gradient bins are chosen resulting in 27 to 117 dimensional descriptors.

For compressing the descriptor, we quantize the gradient histogram in each cell individually and map it to an index. The indices are encoded with fixed length or entropy codes, and the bitstream is concatenated together to form the final descriptor. Fixed-length encoding provides the benefit of compressed domain matching at the cost of a small performance hit. In prior work [12, 1], we have explored several schemes that work well for histogram compression: Huffman Coding, Type Coding and optimal Lloyd-Max VQ. Here, we use Type Coding, which is linear in complexity to the number of histogram bins and performs close to optimal Lloyd-Max VQ. Readers are referred to [1] for details of the histogram quantization and compression schemes.

4 Results

In Section 4.1, we evaluate the different compression schemes at the feature level using Receiver Operating Characteristic (ROC) curves. In Section 4.2, we compare transmitting CHoG descriptors to SIFT descriptors or JPEG images over a 3G network in a real-world mobile visual application.

4.1 Feature Level Performance

For evaluating the performance of low bitrate descriptors, we use the two data sets provided by Winder and Brown in their most recent work [14], *Notre Dame* and *Liberty*. For algorithms that require training, we use the *Notre Dame* data set, while we perform our testing on the *Liberty* set. From the distances between matching and non-matching pairs of descriptors, we obtain a Receiver Operating Characteristic (ROC) curve which plots correct match fraction against incorrect match fraction. For a fair comparison at the same bitrate, we consider the Equal Error Rate (EER) point on the different ROC curves for each scheme.

Figure 2 and Table 1 summarize the bitrate EER trade-off for different schemes. The number of bits required to match the performance of 1024-bit SIFT is presented in Table 1. For Table 1, note that complexity refers to the number of operations required for compressing each descriptor.

First, we compare the 3 hashing schemes. We note that LSH requires about 1000 bits to match the performance of SIFT, which is close to the size of the uncompressed descriptor itself. SSC and Spectral Hashing perform better than LSH at low bitrates but suffer due to overtraining at high bitrates. At high rates, there is a significant gap in performance between the uncompressed 1024-bit SIFT descriptor and hashing schemes based on machine learning. An advantage of hashing schemes is that they can be compared in the compressed domain using look-up tables.

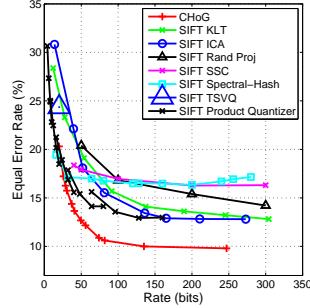


Fig. 2. Comparison of EER versus bitrate for different SIFT compression schemes for the *Liberty* data set. We observe that CHoG outperforms all other schemes.

For transform coding, we observe that the KLT scheme matches the performance of SIFT at about 200 bits. The ICA transform scheme gives a 10-25% reduction in bitrate at a fixed EER compared to the KLT scheme. The transform coding schemes outperform hashing schemes by a significant margin.

The TSVQ compression at 20 bits performs poorly and does not come close to the performance of SIFT. The PQ scheme performs best at low rates. The PQ scheme requires about 160 bits to match the performance of SIFT as also observed by the authors in [10]. Both ICA and PQ schemes require a bitrate of 160 bits to match the performance of SIFT. Note, however, for PQ at this bitrate, the size of the codebook $C=1024$, and the scheme is an order of magnitude more complex than transform coding.

Finally, we observe that CHoG outperforms all SIFT compression schemes. Note from Table 1 that CHoG provides several key advantages: no training, significantly lower complexity $O(d)$, and compressed domain matching. The gain in performance comes from using a more compact spatial footprint, KL distance for comparisons and a highly efficient quantization and compression scheme. We conclude that we can achieve better performance with CHoG, which is designed taking compression into account, compared to compressing SIFT.

Scheme	# of bits	Training	Complexity	CDM
LSH	1000	✗	$O(Nd)$	✓
SSC	-	✓	$O(Nd)$	✓
S-Hash	-	✓	$O(Nd)$	✓
KLT	200	✓	$O(d^2)$	✗
ICA	160	✓	$O(d^2)$	✗
PQ	160	✓	$O(Cd)$	✓
TSVQ	-	✓	$O(BDd)$	✓
CHoG	60	✗	$O(d)$	✓

Table 1. Results for different compression schemes. CDM is Compressed Domain Matching. N is the number of hash-bits. $d = 128$ for SIFT schemes. C = size of codebook for PQ scheme. B = breadth of TSVQ. D = depth of TSVQ.

4.2 Image Retrieval Performance

We evaluate the performance of CHoG in a large scale mobile image retrieval system. We use a database of one million CD, DVD and book cover images, and a set of 1000 query images [15] exhibiting challenging photometric and geometric distortions. The server retrieval pipeline is based on techniques proposed in [9]. More details can be obtained in [1]. Each image has 500×500 pixels resolution. We define Classification Accuracy (CA) as the percentage of query images correctly retrieved. The data transmission experiments are conducted in a AT&T 3G wireless network, averaged over several days, with a total of more than 5000 transmissions at indoor locations where a image-based retrieval system would be typically used.

Figure 3 compares schemes based on CHoG, SIFT and JPEG. For the JPEG scheme, the bitrate is varied by changing the quality of compression. For SIFT, we transmit uncompressed 1024-bit descriptors, and for CHoG, we transmit 60-bit descriptors. For SIFT and CHoG, we sweep the CA-bitrate curve by varying the number of descriptors transmitted. In Figure 3(*left*), we observe that the amount of data for CHoG descriptors are an order of magnitude smaller than JPEG images or SIFT descriptors, to achieve the same CA. In Figure 3(*right*), we study the average end-to-end latency at the highest accuracy point for the different schemes. We achieve approximately 2-4 \times reduction in system latency with CHoG descriptors compared to JPEG images or uncompressed SIFT descriptors.

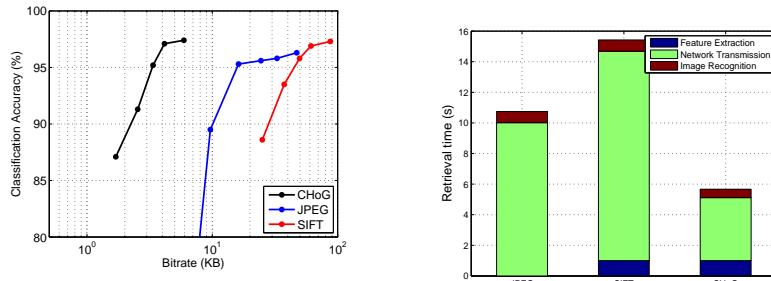


Fig. 3. Figure(*left*) compares bitrate-classification accuracy of different schemes. CHoG descriptor data are an order of magnitude smaller compared to transmitting JPEG images or uncompressed SIFT descriptors, at the same CA. Figure(*right*) compares end-to-end latency for different schemes. Compared to SIFT and JPEG schemes, CHoG achieves approximately 2-4 \times reduction in average system latency in a 3G network.

5 Conclusion

We perform a comprehensive survey of SIFT compression schemes and evaluate them in a common framework. We achieve better performance with CHoG, which is designed taking compression into account, compared to compressing SIFT. The CHoG descriptor at 60 bits matches the performance of the uncompressed 1024-bit SIFT descriptor. We evaluate the performance of CHoG in a large-scale mobile image retrieval system, and show that we can achieve 2-4 \times reduction

in latency by transmitting CHoG descriptors compared to SIFT descriptors or JPEG compressed images.

References

1. Chandrasekhar, V., Reznik, Y., Takacs, G., Chen, D.M., Tsai, S.S., Grzeszczuk, R., Girod, B.: Study of Quantization Schemes for Low Bitrate CHoG descriptors. In: Proceedings of IEEE International Workshop on Mobile Vision (IWMV), San Francisco, California (June 2010)
2. Takacs, G., Chandrasekhar, V., Gelfand, N., Xiong, Y., Chen, W., Bismarck, T., Grzeszczuk, R., Pulli, K., Girod, B.: Outdoors augmented reality on mobile phone using loxel-based visual feature organization. In: Proc. of ACM International Conference on Multimedia Information Retrieval (ACM MIR), Vancouver, Canada (October 2008)
3. Lowe, D.: Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* **60**(2) (2004) 91–110
4. Yeo, C., Ahammad, P., Ramchandran, K.: Rate-efficient visual correspondences using random projections. In: Proc. of IEEE International Conference on Image Processing (ICIP), San Diego, California (October 2008)
5. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM* **51**(1) (2008) 117–122
6. Torralba, A., Fergus, R., Weiss, Y.: Small Codes and Large Image Databases for Recognition. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2008)
7. Weiss, Y., Torralba, A., Fergus, R.: Spectral Hashing. In: Proceedings of Neural Information Processing Systems (NIPS), Vancouver, BC, Canada (December 2008)
8. Chandrasekhar, V., Takacs, G., Chen, D.M., Tsai, S.S., Girod, B.: Transform coding of feature descriptors. In: Proc. of Visual Communications and Image Processing Conference (VCIP), San Jose, California (January 2009)
9. Nistér, D., Stewénius, H.: Scalable recognition with a vocabulary tree. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), New York, USA (June 2006)
10. Jegou, H., Douze, M., Schmid, C.: Product Quantization for Nearest Neighbor Search. Accepted to IEEE Transactions on Pattern Analysis and Machine Intelligence (2010)
11. Jegou, H., Douze, M., Schmid, C.: Hamming embedding and weak geometric consistency for large scale image search. In: Proc. of European Conference on Computer Vision (ECCV), Berlin, Heidelberg (2008) 304–317
12. Chandrasekhar, V., Takacs, G., Chen, D.M., Tsai, S.S., Grzeszczuk, R., Girod, B.: CHoG: Compressed Histogram of Gradients - A low bit rate feature descriptor. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Miami, Florida (June 2009)
13. Narozny, M., Barret, M., Pham, D.T.: Ica based algorithms for computing optimal 1-d linear block transforms in variable high-rate source coding. *Signal Process.* **88**(2) (2008) 268–283
14. Winder, S., Hua, G., Brown, M.: Picking the best daisy. In: Proc. of Computer Vision and Pattern Recognition (CVPR), Miami, Florida (June 2009)
15. Chen, D.M., Tsai, S.S., Vedantham, R., Grzeszczuk, R., Girod, B.: CD Cover Database - Query Images. (April 2008)