International Journal on Computer Vision: Special Issue on Mobile Vision manuscript No.
(will be inserted by the editor)

Compressed Histogram of Gradients: A Low-Bitrate Descriptor.

Vijay Chandrasekhar · Gabriel Takacs · David M. Chen · Sam S. Tsai · Yuriy Reznik · Radek Grzeszczuk · Bernd Girod

Received: date / Accepted: date

Abstract Establishing visual correspondences is an essential component of many computer vision problems, which is often done with local feature-descriptors. Transmission and storage of these descriptors are of critical importance in the context of mobile visual search applications. We propose a framework for computing low bit-rate feature descriptors with a 20× reduction in bit rate compared to state-of-the-art descriptors. The framework offers low complexity and has significant speed-up in the matching stage. We show how to efficiently compute distances between descriptors in the compressed domain eliminating the need for decoding. We perform a comprehensive performance comparison with SIFT, SURF, BRIEF, MPEG-7 image signatures and other low bit-rate descriptors and show that our proposed CHoG descriptor outperforms existing schemes significantly over a wide range of bitrates. We implement the descriptor in a mobile image retrieval system and for a database of 1 million CD, DVD and book covers, we achieve 96% retrieval accuracy using only 4 KB of data per query image.

Keywords CHoG · feature descriptor · mobile visual search · content-based image retrieval · histogram-of-gradients · low bitrate

This work was first presented as an oral presentation at Computer Vision and Pattern Recognition (CVPR), 2009. Since then, the authors have studied feature compression in more detail in (Chandrasekhar et al., 2009a, 2010a,c,b). A default implementation of CHoG is available at http://www.stanford.edu/ vijayc/

Vijay Chandrasekhar
Stanford University
Tel. : +1-650-723-3476
Fax: +1-650-724-3648
E-mail: vijayc@stanford.edu

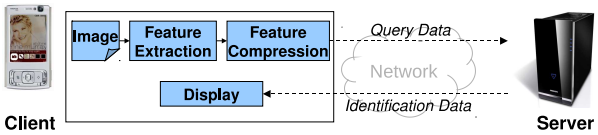


Fig. 1 Example of a mobile visual search application. The user points his camera phone at an object and obtains relevant information about it. Feature compression is key to achieving low system latency. By transmitting compressed descriptors from the mobile-phone, one can reduce system latency significantly.

1 Introduction

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 equipped with GPS, 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 (Fig 1). Such applications can be used, e.g., for identifying products, comparison shopping, finding information about movies, CDs, real estate, print media or artworks. First commercial deployments of such systems include Google Goggles (Google, 2009), Nokia Point and Find (Nokia, 2006), Kooaba (Kooaba, 2007), Ricoh iCandy (Erol et al., 2008; Graham and Hull, 2008; Hull et al., 2007) and Snaptell (Amazon, 2007).

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 JPEG image could take tens of seconds over a slow wireless link. On the other hand, extraction of salient image features is now possible on mobile devices in seconds or less. 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.
- 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 feature data from the server, and all image matching is performed on the device.

When the database is small, it can be stored on the phone and image retrieval algorithms can be run locally (Takacs et al., 2008). When the database is large, it has to be placed on a remote server and retrieval algorithms are run remotely. In each case, feature compression is key to decreasing the amount of the data transmitted, and thus, reducing network latency. A small descriptor also helps if the database is stored on the mobile device. The smaller the descriptor, the more features can be stored in limited memory.

Since Lowe's paper in 1999 (Lowe, 1999), 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) (Mikolajczyk and Schmid, 2005), Speeded Up Robust Features (SURF) (Bay et al., 2008), and the machine-optimized gradient-based descriptors (Winder and Brown, 2007; Winder et al., 2009). As a 128-dimensional descriptor, SIFT is conventionally stored as 1024 bits (8 bits/dimension). Alas, the size of SIFT descriptor data from an image is typically larger than the size of the JPEG compressed image itself, making it unsuitable for mobile applications.

Several compression schemes have been proposed to reduce the bitrate of SIFT descriptors. In our recent work (Chandrasekhar et al., 2010b), we survey different SIFT compression schemes. They can be broadly categorized into schemes based on hashing (Yeo et al., 2008; Torralba et al., 2008; Weiss et al., 2008), transform coding (Chandrasekhar et al., 2009a, 2010b) and vector quantization (Jegou et al., 2010; Nistér and Stewénius, 2006; Jegou et al., 2008). We note that hashing schemes like Locality Sensitive Hashing (LSH), Similarity Sensitive Coding (SSC) or Spectral Hashing (SH) do not perform well at low bitrates. Conventional transform coding schemes based on Principal Component Analysis (PCA) do not work well due to the highly non-Gaussian statistics of the SIFT descriptor. Vector quantization schemes based on the Product Quantizer (Jegou et al., 2010) or a Tree Structured Vector Quantizer (Nistér and Stewénius, 2006) are complex and require storage of large codebooks on the mobile device.

Other popular approaches used to reduce the size of descriptors typically employ dimensionality reduction via PCA or Linear Discriminant Analysis (LDA) (Ke and Sukthankar, 2004; Hua et al., 2007). Ke et al. (Ke and Sukthankar, 2004) investigate dimensionality reduction of patches directly via PCA. Hua et al. (Hua et al., 2007) propose a scheme that uses LDA. Winder and Brown (Winder et al., 2009) 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 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 coding is employed. The 60-bit MPEG-7 Trace Transform descriptor (Brasnett and Bober, 2007), Transform coded SURF features Chandrasekhar et al. (2009a) and Binary Robust Independent Elementary Features (BRIEF) (Calonder et al., 2010) are other examples of low-bitrate descriptors proposed in recent literature. Johnson proposes a generalized set of techniques to compress local features in his recent work Johnson (2010).

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 at low bitrates. Ideally, it would permit descriptor comparisons in the compressed domain for speedy feature matching. Further, we would like to avoid a training step so that the descriptor is not dependent on any specific data set. Finally, the compression algorithm should have low complexity so that it can be efficiently implemented on mobile devices. To meet all these requirements simultaneously, we designed the Compressed Histogram of Gradients (CHoG) descriptor (Chandrasekhar et al., 2009b, 2010c).

The outline of the paper is as follows. In Section 2, we review Histogram-of-Gradients (HoG) descriptors, and discuss the design of the CHoG descriptor. We discuss different quantization and compression schemes used to generate low bitrate CHoG descriptors. In Section 3, we perform a comprehensive survey of several low bitrate descriptors proposed in the literature and show that CHoG outperforms all schemes. We present both feature-level results and image-level retrieval results in a practical mobile product search system.

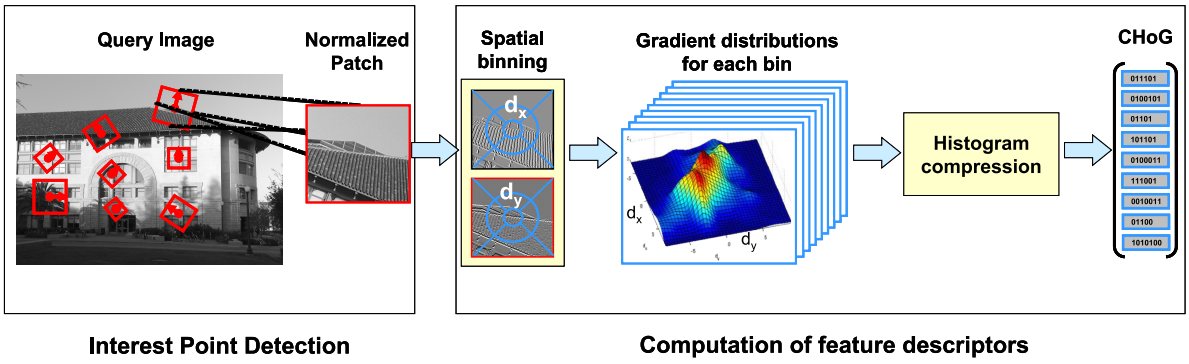


Fig. 2 Illustration of CHoG feature descriptors. We first start with patches obtained from interest points (e.g., corners, blobs) at different scales. The patches at different scales are oriented along the dominant gradient. We first divide the scaled and oriented canonical patches into log-polar spatial bins. Then, we perform independent quantization of histograms in each spatial bin. The resulting codebook indices are then encoded using fixed-length or arithmetic codes. The final bitstream of the feature descriptor is formed as a concatenation of codes representative of histograms in each spatial bin. CHoG descriptors at 60 bits match the performance of 1024-bit SIFT descriptors.

2 Descriptor Design

The goal of a feature descriptor is to robustly capture salient information from a canonical image patch. We use a histogram-of-gradients descriptor and explicitly exploit the anisotropic statistics of the underlying gradient distributions. By directly capturing the gradient distribution, we can use more effective distance measures like Kullback-Leibler (KL) divergence, and more importantly, we can apply quantization and compression schemes that work well for distributions to produce compact descriptors. In Section 2.2, we discuss the choice of parameters of our Uncompressed Histogram of Gradients (UHoG) descriptor. In Section 2.3, we discuss quantization and compression schemes that enable low bitrate Compressed Histogram of Gradient (CHoG) descriptors. First, in Section 2.1, we describe the framework used for evaluating descriptors.

2.1 Descriptor Evaluation

For evaluating the performance of low bitrate descriptors, we use the two data sets provided by Winder et al. in their most recent work (Winder et al., 2009), *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. We use the methodology proposed in Winder et al. (Winder et al., 2009) for evaluating descriptors. We compute a distance between each matching and non-matching pair of descriptors. The distance measure used depends on the descriptor. For example, CHoG descriptors use the symmetric Kullback-Leibler (KL) (Cover and Thomas, 2006) as it performs better than L_1 or L_2 norm for comparing histograms (Chandrasekhar et al., 2009b). From these distances, we obtain a Receiver Operating Characteristic (ROC) curve which plots correct match fraction against incorrect match fraction. We show the performance of the 1024-bit SIFT descriptor in each ROC plot. Our focus is on descriptors that perform on par with SIFT and are in the range of 50-100 bits.

2.2 Histogram-of-Gradient Based Descriptors

A number of different feature descriptors are based on the distribution of gradients within an image patch: Lowe (Lowe, 2004), Bay et al. (Bay et al., 2008), Dalal and Triggs (Dalal and Triggs, 2005), Freeman and Roth (Freeman and Roth, 1994), Winder and Brown (Winder et al., 2009). In this section, we describe the pipeline used to compute gradient histogram descriptors, and then show the relationships between SIFT, SURF and our proposed descriptor.

The CHoG descriptor pipeline is illustrated in Fig. 2. As in (Mikolajczyk et al., 2005), we model illumination changes to the patch appearance by a simple affine transformation, $aI + b$, of the pixel intensities, which is compensated by normalizing the mean and standard deviation of the pixel values of each patch. Next, we apply an additional Gaussian smoothing of $\sigma = 2.7$ pixels to the patch. The smoothing parameter is obtained as the optimal value from the learning algorithm proposed by Winder and Brown, for the data sets in consideration. Local image gradients d_x and d_y are computed using a centered derivative mask $[-1, 0, 1]$. Next, the patch is divided into localized spatial bins. The granularity of

spatial binning is determined by a tension between discriminative power and robustness to minor variations in interest point localization error. Then, some statistics of d_x and d_y are extracted separately for each spatial bin, forming the UHoG descriptor.

SIFT and SURF descriptors can be calculated as functions of the gradient histograms, provided that such histograms are available for each spatial bin and the d_x, d_y values are sorted into sufficiently fine bins. Let $P_{D_x,D_y}(d_x, d_y)$ be the normalized joint (x, y)-gradient histogram in a spatial bin. Note that the gradients within a spatial bin may be weighted by a Gaussian window prior to descriptor computation (Lowe, 2004; Bay et al., 2006).

The 8 SIFT components of a spatial bin, $\mathcal{D}_{SIFT}$, are

$$\mathcal{D}_{SIFT}(i) = \sum_{(d_x,d_y)\,\in\,\Omega_i} \sqrt{d_x^2 + d_y^2}\; P_{D_x,D_y}(d_x, d_y) \qquad (1)$$

where $\Omega_i = \{\ (d_x, d_y)\ |\ \frac{\pi(i-1)}{4} \leq \tan^{-1}\frac{d_y}{d_x} < \frac{\pi i}{4}\ ,\ i = 1 \ldots 8\}$. Similarly, the 4 SURF components of a spatial bin, $\mathcal{D}_{SURF}$, are

$$\mathcal{D}_{SURF}(1) = \sum_{d_x}\sum_{d_y} P_{D_x,D_y}(d_x, d_y)|d_x| \qquad (2)$$

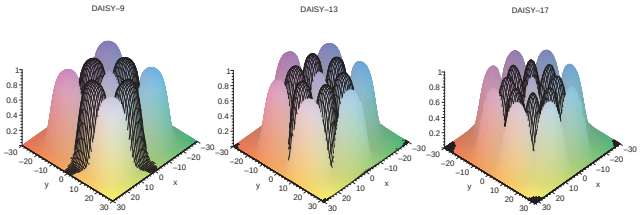
$$\mathcal{D}_{SURF}(2) = \sum_{d_x}\sum_{d_y} P_{D_x,D_y}(d_x, d_y)\ d_x \qquad (3)$$

$$\mathcal{D}_{SURF}(3) = \sum_{d_x}\sum_{d_y} P_{D_x,D_y}(d_x, d_y)|d_y| \qquad (4)$$

$$\mathcal{D}_{SURF}(4) = \sum_{d_x}\sum_{d_y} P_{D_x,D_y}(d_x, d_y)\ d_y \qquad (5)$$

For CHoG, we propose coarse quantization of the 2D gradient histogram, and encoding the histogram directly as a descriptor. We approximate $P_{D_x,D_y}(d_x, d_y)$ as $\hat{P}_{\hat{D}_x,\hat{D}_y}(\hat{d}_x, \hat{d}_y)$ for $(\hat{d}_x, \hat{d}_y)$ $\in S$, where S represents a small number of quantization centroids or bins as shown in Fig. 4. We refer to this uncompressed descriptor representation $\hat{P}_{\hat{D}_x,\hat{D}_y}$ (for all spatial bins) as Uncompressed Histogram of Gradients (UHoG), which is obtained by counting the number of pixels which get quantized to each centroid in S, and then L_1 normalized.

The i^{th} UHoG descriptor is defined as $\mathcal{D}^i_{UHoG} = \left[\hat{P}^i_1, \hat{P}^i_2, \ldots, \hat{P}^i_N\right]$, where $\hat{P}^i_k$ represents the gradient histograms in spatial bin k of descriptor i, and N is the total number of spatial bins. Note that the dimensionality of UHoG is given by $N \times B$, where N is the number of spatial bins, and B is the number of bins in the gradient histogram. Next, we discuss the parameters chosen for spatial and gradient binning.



(*DAISY-9*) (*DAISY-13*) (*DAISY-17*)

Fig. 3 DAISY configurations with $K = 9, 13, 17$ spatial bins. We use Gaussian-shaped overlapping (soft) binning.

2.2.1 Spatial Binning

Since we want a very compact descriptor, we have experimented with reducing the number of spatial bins. Fewer spatial bins means fewer histograms and a smaller descriptor. However, it is important that we do not adversely affect the performance of the descriptor. SIFT and SURF use a square 4×4 grid with 16 cells. We divide the patch into log-polar configurations as proposed in (Tola et al., 2008; Mikolajczyk and Schmid, 2005; Winder et al., 2009). The log-polar configurations have been shown to perform better than the 4×4 square-grid spatial binning used in SIFT (Winder et al., 2009). There is one key difference between the DAISY configurations proposed in (Winder et al., 2009), and the configurations shown in Fig. 3. In (Winder et al., 2009), the authors divide the patch into disjoint localized cells. We use overlapping regions for spatial binning (Fig. 3) which improves the performance of the descriptor by making it more robust to interest point localization error. The soft assignment is made such that each pixel contributes to multiple spatial bins with normalized Gaussian weights that sum to 1. A value of σ for the Gaussian that works well is $d_{min}/3$, where d_{min} is the minimum distance between bin centroids in the DAISY configuration. Intuitively, a pixel close to a bin centroid should contribute little to other spatial bins. The DAISY-9 configuration matches the performance of the 4×4 square-grid configuration, and hence, we use it for all the experiments in this section. Next, we discuss how the gradient binning is done.

2.2.2 Gradient Histogram Binning

As stated earlier, we wish to approximate the histogram of gradients with a small set of bins, S. We propose histogram binning schemes that exploit the underlying gradient statistics observed in patches extracted around interest points. The joint distribution of (d_x, d_y) for 10000 cells from the training data set is shown in Fig. 4(a,b). We observe that the distribution is strongly peaked around $(0, 0)$, and that the variance is higher

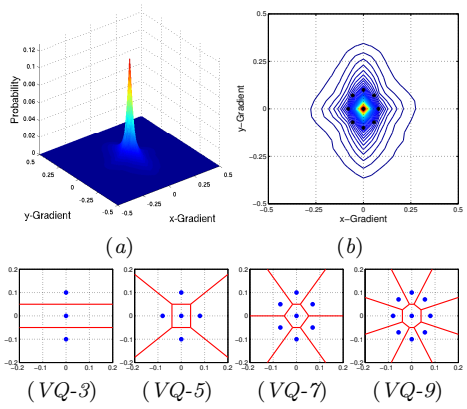


Fig. 4 The joint (d_x, d_y) gradient distribution (a) over a large number of cells, and (b), its contour plot. The greater variance in y-axis results from aligning the patches along the most dominant gradient after interest point detection. The quantization bin constellations VQ-3, VQ-5, VQ-7 and VQ-9 and their associated Voronoi cells are shown at the bottom.

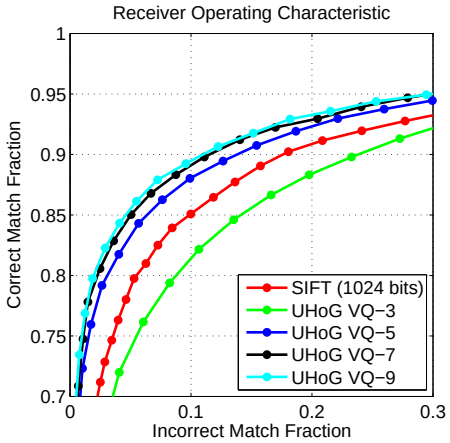


Fig. 5 ROC curves for various gradient binning configurations. DAISY-9 spatial bin configuration and symmetric KL divergence are used. The VQ-5 configuration matches the performance of SIFT.

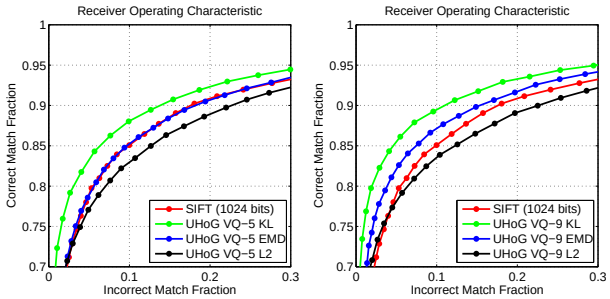


Fig. 6 ROC curves for distance measures for gradient-bin configurations VQ-5 (b) and VQ-9 (b), and spatial-bin configuration DAISY-9. KL and EMD consistently outperform the conventional L_2-norm used for comparing descriptors.

for the y-gradient. This anisotropic distribution is a result of canonical image patches being oriented along the most dominant gradient by the interest point detector.

We perform a vector quantization of the gradient vectors into a small set of bin centers, S, shown in Fig. 4. We call these bin configurations VQ-3, VQ-5, VQ-7 and VQ-9. All bin configurations have a bin center at $(0, 0)$ to capture the central peak of the gradient distribution. The additional bin centers are evenly spaced (with respect to angle) over ellipses, the eccentricity of which are chosen in accordance with the observed skew in the gradient statistics. Similar to soft spatial binning, we assign each (d_x, d_y) pair to multiple bin centers with normalized Gaussian weights. Again, we use $\sigma = q_{min}/3$, where q_{min} is the minimum distance between centroids in the VQ bin configurations shown in Fig. 4.

To evaluate the performance of each gradient-binning configuration, we plot the ROC curves in Fig. 5. As we increase the number of bin centers, we obtain a more accurate approximation of the gradient distribution, and the performance of the descriptor improves. We observe that the VQ-5 and DAISY-9 UHoG configuration suffices to match the performance of SIFT.

2.2.3 Distance Measures

Since UHoG is a direct representation of a histogram we can use distance measures that are well-suited to histogram comparison. Several quantitative measures have been proposed to compare distributions in the literature. We consider three measures, the L_2-norm, Kullback-Leibler divergence (Kullback, 1987), and the Earth Mover's Distance (EMD) (Rubner et al., 2000).

The distance between two UHoG (or CHoG) descriptors is defined as $d(\mathcal{D}^i, \mathcal{D}^j) = \sum_{k=1}^{N} d_{\text{hist}}(\hat{P}^i_k, \hat{P}^j_k)$, where N is the number of spatial bins, d_{hist} is a distance measure between two distributions, and $\hat{P}^i$ represents the gradient distribution in a spatial bin.

Let B denote the number of bins in the gradient histogram, and $\hat{P}^i = [p^i_1, p^i_2 .... p^i_B]$. We define d_{KL} as the symmetric KL divergence between two histograms such that,

$$d_{\text{KL}}(\hat{P}^i, \hat{P}^j) = \sum_{n=1}^{B} p^i_n \log \frac{p^i_n}{p^j_n} + \sum_{n=1}^{B} p^j_n \log \frac{p^j_n}{p^i_n}. \qquad (6)$$

The EMD is a cross-bin histogram distance measure, unlike L_2-norm and KL divergence which are bin-by-bin distance measures. The EMD is the minimum cost to transform one histogram into the other, where there is a "ground distance" defined between each pair of bins. This "ground distance" is the distance between the bin-centers shown in Fig. 4. Note that EMD is a

metric and observes the triangle inequality, while KL divergence is not.

In Fig. 6 we plot ROC curves for different distance measures for VQ-5 and VQ-9. The KL divergence and EMD consistently outperform the L_2-norm, with KL divergence performing the best. Further, KL divergence, being a bin-by-bin measure, is a lot less complex to compute than the EMD. For this reason, we use the KL divergence as the distance measure for all the CHoG experiments in this paper. Next, this observation motivates techniques to compress probability distributions which minimize distortion in KL divergence.

2.3 Histogram Quantization and Compression

Our goal is to produce low bit-rate Compressed Histogram of Gradients (CHoG) descriptors while maintaining the highest possible recognition performance. Lossy compression of probability distributions is an interesting problem that has not received much attention in the literature.

In this section, we discuss three different schemes for quantization and compression of distributions: Huffman Coding, Type Coding and Entropy Constrained Vector Quantization (ECVQ). We note that ECVQ can achieve optimal rate-distortion performance and thus provide a bound on performance of other schemes. However, ECVQ requires expensive training with the generalized Lloyd algorithm, and requires the storage of unstructured codebooks on the mobile device for compression. For mobile applications, the compression scheme should require a small amount of memory and have low computational complexity. As we will see, the two proposed schemes, Huffman Coding and Type Coding, come close to achieving the performance of optimal ECVQ, while being of much lower complexity, and do not require explicit storage of codebooks on the client.

Let m represent the number of gradient bins. Let $P = [p_1, p_2, ...p_m] \in R_+^m$ be the original normalized histogram, and $Q = [q_1, q_2, ...q_m] \in R_+^m$ be the quantized normalized histogram defined over the same sample space. As mentioned earlier, we are primarily interested in the symmetric KL divergence as a distance measure.

For each scheme, we quantize the gradient histogram in each cell individually and map it to an index. The indices are then encoded with either a fixed-length code or variable-length code. The codewords are concatenated to form the final descriptor. We also experimented with joint coding of the gradient histograms in different cells, but this did not yield any practical gain. Next, for each compression scheme, we discuss the quantization theory and implementation details, illustrate an example and present ROC results benchmarked against SIFT. Finally, we compare the performance of the different schemes in a common framework.

2.3.1 Huffman Tree Coding

Given a probability distribution, one way to compress it is to construct and store a Huffman tree built from the distribution (Gagie, 2006; Chandrasekhar et al., 2009b). From this tree, the Huffman codes, $\{c_1, \ldots, c_n\}$, of each symbol in the distribution are computed. The reconstructed distribution, Q, can be subsequently obtained as $q_i = 2^{-b_i}$, where b_i is the number of bits in c_i. It is well known that Huffman tree coding guarantees that $D(P \,||\, Q) < 1$, where $D(P \,||\, Q) = \sum_{i=1}^{n} p_i \log_2 \frac{p_i}{q_i}$ (Cover and Thomas, 2006).

Huffman trees are strict binary trees, such that each node has exactly zero or two children. The maximum depth of a strict binary tree with n leaf nodes is $n-1$. Therefore, a Huffman tree can be stored in $(n-1)\lceil \log(n-1) \rceil$ bits by storing the depth of each symbol in the Huffman tree with a fixed length code. The depth of the last leaf node does not need to be stored, since a Huffman tree is a strict binary tree and $\sum q_i = 1$. We call this scheme Tree Depth Coding (TDC). It was proposed in (Gagie, 2006).

While TDC can be used for all m, we show how to reduce the bit rate further for small m in (Chandrasekhar et al., 2009b). We reduce the bits needed to store a tree by enumerating all possible trees, and using fixed-length codes to represent them. The number of Huffman trees $T(m)$ utilized by such a scheme can be estimated by considering labeling of all possible rooted binary trees with m leaves

$$T(m) < m!\, C_{m-1}\,, \tag{7}$$

where $C_n = \frac{1}{n+1}\binom{2n}{n}$ is the Catalan number. Hence, the index of a Huffman tree representing distribution P with fixed-length encoding requires at most

$$R_{\mathrm{Huf}}(m) \leq \lceil \log_2 T(m) \rceil \sim m \log_2 m + O(m)\,. \tag{8}$$

bits to encode. For some small values of m, we can achieve further compression by entropy coding the fixed-length tree indices. This is because not all trees are equally likely to occur from gradient statistics. We refer to the fixed and variable bitrate tree enumeration schemes as the Tree Fixed Length Coding and the Tree Entropy Coding respectively.

Implementation. Quantization is implemented by a standard Huffman tree construction algorithm, requiring $O(m \log m)$ operations, where m is the number of bins in the gradient histogram. All unique Huffman

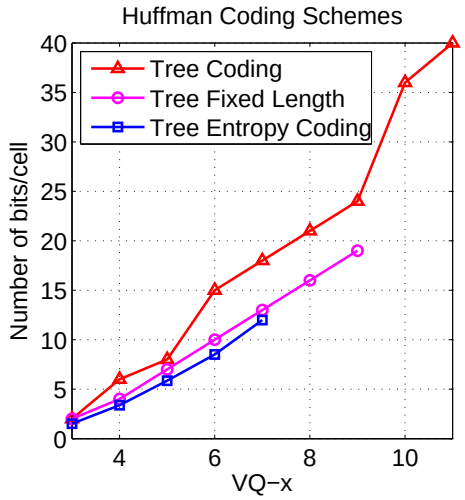


Fig. 7 Number of bits/spatial bin for Huffman histograms using different schemes. Note that the same Huffman quantization scheme is applied for all three schemes before encoding. We can obtain 25-50% compression compared to the Tree Depth Coding representation. Note the ranges of m in which Tree Fixed Length and Tree Entropy Coding can be used.

trees are enumerated and their indices are stored in memory. The number of Huffman trees for $m = 3, 5, 7, 9$ are $3, 75, 4347$ and 441675 respectively. The number of trees grows very rapidly with m and tree enumeration becomes impractical beyond $m = 9$. For $m \leq 7$, we found entropy coding to be useful, resulting in savings of $10-20\%$ in the bitrate. This compression is achieved by using a context-adaptive binary arithmetic coding. In Fig. 7, we show that we can obtain 25-50% compression compared to the naive Tree Depth Coding scheme.

Example. Let $m = 5$ corresponding to the VQ-5 gradient bin configuration. Let $P = [0.1, 0.3, 0.2, 0.25, 0.15]$ be the original distribution as described by the histogram. We build a Huffman tree on P, and thus quantize the distribution to $Q = [0.125, 0.25, 0.25, 0.25, 0.125]$. The quantized distribution Q is thus mapped to one of 75 possible Huffman trees with $m = 5$ leave nodes. It can be communicated with a fixed length code of $\lceil \log_2 75 \rceil = 7$ bits.

ROC Results. Figure 8 shows the performance of the Huffman compression scheme for the DAISY-9 configuration. The bitrate in Figure 8 is varied by increasing the number of gradient bins from 5 to 9. For the DAISY-9, VQ-7 configuration, the descriptor at 88 bits outperforms SIFT at 1024 bits.

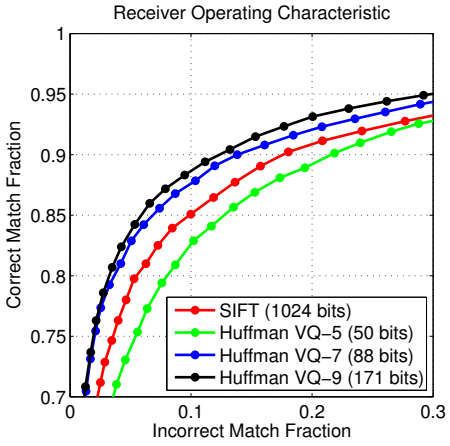


Fig. 8 ROC curves for compressing distributions with Huffman scheme for the DAISY-9 configuration for the *Liberty* data set. The CHoG descriptor at 88 bits outperforms SIFT at 1024 bits.

2.3.2 Type Quantization

The idea of type coding is to construct a lattice of distributions (or *types*) $Q = Q(k_1, \ldots, k_m)$ with probabilities

$$q_i = \frac{k_i}{n}, \quad k_i,\, n \in Z_+, \quad \sum_i k_i = n \tag{9}$$

and then pick and transmit the index of the type that is closest to the original distribution P (Chandrasekhar et al., 2010c; Reznik et al., 2010). The parameter n is used to control the number/density of reconstruction points.

We note that type coding is related to the A_n lattice (Conway and Sloane, 1982). The distinctive part of our problem is the particular shape of the set that we need to quantize. The type lattice is naturally defined within a bounded subset of the R^m space, which is the unit $(m-1)$-simplex, compared to the conventional m-dimensional unit cube. This is precisely the space containing all possible input probability vectors. We show examples of type lattices constructed for $m = 3$ and $n = 1, \ldots, 3$ in Figure 9.

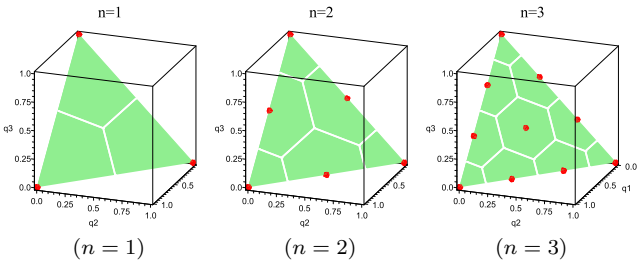


Fig. 9 Type lattices and their Voronoi partitions in 3 dimensions $(m = 3, n = 1, 2, 3)$.

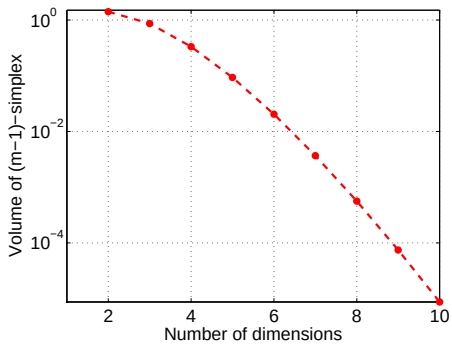


Fig. 10 Volume of $m-1$-simplex compared to the m dimensional unit cube of volume 1. We note that the volume rapidly decreases as the number of bins increases.

The volume of the $(m-1)$-simplex that we need to quantize is given by (Sommerville, 1958)

$$V_{m-1} = \frac{a^{m-1}}{(m-1)!}\sqrt{\frac{m}{2^{m-1}}}\,\Big|_{a=\sqrt{2}} = \frac{\sqrt{m}}{(m-1)!}\,. \tag{10}$$

In Figure 10, we note that the volume is rapidly decaying as the number of bins m increases. As a result, we expect type coding to become increasingly efficient compared to lattice quantization over the entire unit cube as m increases. For a detailed discussion of rate-distortion characteristics, readers are referred to (Reznik et al., 2010).

The total number of types in lattice (9) is essentially the number of partitions of parameter n into m terms $k_1 + \ldots + k_m = n$, given by a *multiset coefficient*:

$$\left(\!\binom{m}{n}\!\right) = \binom{n+m-1}{m-1}\,. \tag{11}$$

Consequently, the rate needed for encoding of types satisfies:

$$R_{Type}(m,n) \le \left\lceil \log_2 \left(\!\binom{m}{n}\!\right) \right\rceil \sim (m-1)\log_2 n\,. \tag{12}$$

Next, we develop a combinatorial enumeration scheme for fast indexing and compressed domain matching of descriptors.

Quantization. In order to quantize a given input distribution P to the nearest type, we use the algorithm described below. This algorithm is similar to Conway and Sloane's quantizer for A_n lattice (Conway and Sloane, 1982), but it works within a bounded subset of R^m.

1. Compute numbers (best unconstrained approximation)

$$k'_i = \left\lfloor np_i + \tfrac{1}{2} \right\rfloor\,, \quad n' = \sum_i k'_i\,.$$

2. If $n' = n$ we are done. Otherwise, compute errors

$$\delta_i = k'_i - np_i\,,$$

and sort them such that

$$-\tfrac{1}{2} \le \delta_{j_1} \le \delta_{j_2} \le \ldots \le \delta_{j_m} < \tfrac{1}{2}\,,$$

3. Let $d = n' - n$. If $d > 0$ then we decrement d values k'_i with largest errors

$$k_{j_i} = \begin{bmatrix} k'_{j_i} & j = 1,\ldots,m-d-1\,, \\ k'_{j_i} - 1 & i = m-d,\ldots,m\,, \end{bmatrix}$$

otherwise, if $d < 0$ we increment $|d|$ values k'_i with smallest errors

$$k_{j_i} = \begin{bmatrix} k'_{j_i} + 1 & i = 1,\ldots,|d|\,, \\ k'_{j_i} & i = |d|+1,\ldots,m\,. \end{bmatrix}$$

Enumeration of types. We compute a unique index $\xi(k_1,\ldots,k_m)$ for a type with coordinates $k_1,\ldots,k_m$ using:

$$\xi(k_1,\ldots,k_n) = \sum_{j=1}^{n-2}\sum_{i=0}^{k_j-1}\left(\!\binom{m-j}{n-i-\sum_{l=1}^{j-1}k_l}\!\right) + k_{n-1}. \tag{13}$$

This formula follows by induction (starting with $m = 2, 3$, etc.), and it implements lexicographic enumeration of types. For example:

$$\begin{aligned} \xi(0,0,\ldots,0,n) &= 0\,, \\ \xi(0,0,\ldots,1,n-1) &= 1\,, \\ &\ldots \\ \xi(n,0,\ldots,0,0) &= \left(\!\binom{m}{n}\!\right) - 1\,. \end{aligned}$$

This direct enumeration allows encoding/decoding operations to be performed without storing any "codebook" or "index" of reconstruction points.

Implementation. We implement enumeration of types according to 13 by using an array of precomputed multiset coefficients. This reduces complexity of enumeration to just about $O(n)$ additions. In implementing type quantization, we observed that the mismatch $d = n' - n$ is typically very small, and so instead of performing full sorting step 2, we simply search for d largest or smallest numbers. With such optimization, the complexity of the algorithm becomes close to $O(m)$, instead of $O(m \log m)$ implied by the use of full search.

We also found it useful to bias type distributions as follows

$$q_i = \frac{k_i + \beta}{n + \beta m}\,. \tag{14}$$

where parameter $\beta \geq 0$ is called the *prior*. The most commonly used values of β in statistics are Jeffrey's prior $\beta = 1/2$, and Laplace prior $\beta = 1$. A value of parameter β that works well is the scaled prior $\beta = \beta_0 \frac{n}{n_0}$, where n_0 is the total number of samples in the original (non-quantized) histogram, and $\beta_0 = 0.5$ is the *prior* used in computation of probabilities P. Finally, for encoding of type indices, we use both fixed-length and entropy coding schemes. We find that entropy coding with an arithmetic coder saves approximately $10-20\%$ in the bitrate. When fixed-length codes are used, we can perform fast compressed domain matching.

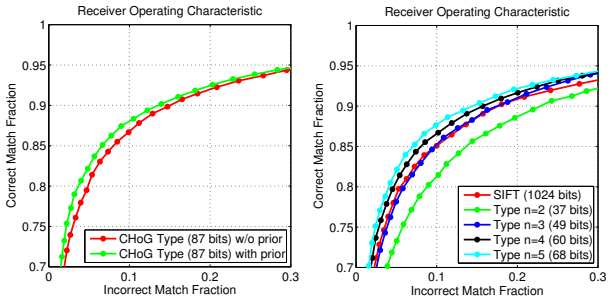
Example. Let $m = 5$, corresponding to the VQ-5 gradient bin configuration. Let the original type described by the histogram be $T = [12, 28, 17, 27, 16]$ and $P = [0.12, 0.28, 0.17, 0.27, 0.16]$ be the corresponding distribution. Let $n = 10$ be the quantization parameter chosen for type coding. The approximation of the type T is $K = [1, 3, 2, 3, 2]$ based on Step (1) of the quantization algorithm. Since $\sum_i k_i \neq 10$, we use the proposed quantization algorithm to obtain quantized type $K = [1, 3, 2, 3, 1]$. The number of samples n_0 in the original histogram is 100, and hence, the scaled prior is computed as $\beta = 0.5 \times 10/100 = 0.05$, and the quantized distribution with prior is $Q = [0.1024, 0.298, 0.2, 0.2976, 0.1024]$. The total number of quantized types is $\binom{14}{4} = 1001$, and Q can be communicated with a fixed length code of $\lceil \log_2 1001 \rceil = 10$ bits.

ROC Results. Figure 11(a) illustrates the advantage of using biased types (14). Figure 11(b) shows performance of the type compression scheme for the DAISY-9, VQ-7 configuration. The bitrate in Figure 11(b) is varied by changing type quantization parameter n. For this configuration, the descriptor at 60 bits outperforms SIFT at 1024 bits.

2.3.3 Entropy Constrained Vector Quantization

We use ECVQ designed with the generalized Lloyd algorithm (Chou et al., 1989) to compute a bound on the performance that can be achieved with the CHoG descriptor framework. The ECVQ scheme is computationally complex, and it is not practical for mobile applications.

The ECVQ algorithm resembles k-means clustering in the statistics community, and, in fact, contains it as a special case. Like k-means clustering, the generalized Lloyd algorithm assigns data to the nearest cluster centers, next computes new cluster centers based on this assignment, and then iterates the two steps until



(a) (b)

Fig. 11 Fig. (a) shows the ROC curves of a type coded CHoG descriptor with and without priors. The performance of the descriptor is better with the scaled prior. Fig. (b) shows ROC curves for compressing distributions with type coding scheme for DAISY-9 and VQ-7 configuration for *Liberty* data set. CHoG descriptor at 60 bits outperforms SIFT at 1024 bits.

convergence is reached. What distinguishes the generalized Lloyd algorithm from k-means is a Lagrangian term which biases the distance measure to reflect the different number of bits required to indicate different clusters. With entropy coding, likely cluster centers will need fewer bits, while unlikely cluster centers require more bits. To properly account for bitrate, cluster probabilities are updated in each iteration of the generalized Lloyd algorithm, much like the cluster centers. We show how the ECVQ scheme can be adapted to the current CHoG framework.

Let $X^m = [p_1, p_2, p_3, ..p_m] \in R_+^m$ denote a normalized histogram. Let P_{X^m} be the distribution of X^m. Let ρ be the distance measure used to compare histograms. Let λ be the Lagrange multiplier. Let ψ be an index set, and let $\alpha : \ X^m \mapsto \psi$ quantize input vectors to indices. Let $\beta : \ \psi \mapsto C$ map indices to a set of centroids $C \in R_+^m$. Let the initial size of the codebook be $K = |\psi|$. Let $\gamma(i)$ be the rate of transmitting centroid i, $i \in \psi$.

The iterative algorithm used is discussed below. The input of the algorithm is a set of points X^m, and the output is the codebook $C = \{\beta(i)\}_{i \in \psi}$. We initialize the algorithm with C as K random points and $\gamma(i) = \log_2(K)$.

1. $\alpha(x^n) = \arg\min_{i \in \psi} \rho(x^n, \beta(i)) + \lambda |\gamma(i)|$
2. $|\gamma(i)| = -\log_2 P_{X^n}(\alpha(X^n) = i)$
3. $\beta(i) = E[X^m | \alpha(X^m) = i]$

We repeat Steps (1)-(3) until convergence. Step (1) is the "assignment step", and Steps (2) and (3) are the "re-estimation steps" where the centroids $\beta(i)$ and rates $\gamma(i)$ are updated. In (Chandrasekhar et al., 2009b), we show that comparing gradient histograms with symmetric KL divergence provides better ROC performance than using L_1 or L_2-norm. It is shown in (Banerjee et al., 2004; Rebollo-Monedero, 2007) that the Lloyd

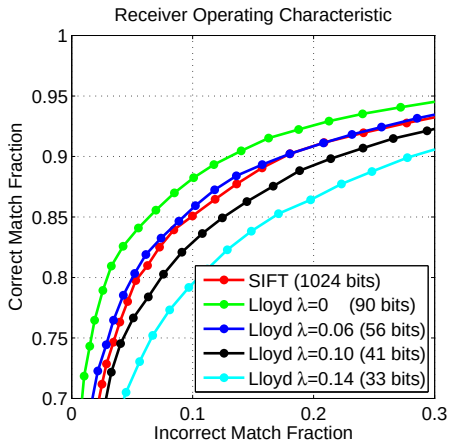


Fig. 12 ROC curves for compressing distributions with Lloyd scheme for DAISY-9 and VQ-7 configuration for the *Liberty* data set. CHoG descriptor at 56 bits outperforms SIFT at 1024 bits.

algorithm can be used for the general class of distance measures called Bregman divergences. Since the symmetric KL-divergence is a Bregman divergence, it can be used as the distance measure in step (1) and the centroid assignment step (3) is nevertheless optimal.

Implementation. We start with an initial codebook size of $K = 1024$ and sweep across λ to vary the bitrate for each gradient configuration. The rate decreases and the distortion increases as we increase the parameter λ. The algorithm itself reduces the size of the codebook as λ increases because certain cells become unpopulated. We add a prior of $\beta_0 = 0.5$ to all bins to avoid singularity problems. Once the histogram is quantized and mapped to an index, we entropy code the indices with an arithmetic coder. Entropy coding typically provides a $10-20\%$ reduction in bitrate compared to fixed length coding. The compression complexity of the scheme is $O(mk)$, where k is the number of cluster centroids and m is the number of gradient bins. Note that this search required to find the representative vector in the unstructured codebook is expensive, and hence, is not suitable for mobile applications.

ROC Results. We show the performance of this scheme in Figure 12 for the DAISY-9, VQ-7 configuration. In Figure 12, the bitrate is varied by increasing λ with an initial codebook size of $K = 1024$. For $\lambda = 0$, we represent the descriptor with fixed-length codes in 90 bits. For this configuration, the descriptor at 56 bits outperforms SIFT at 1024 bits. Next, we compare the performance of the different histogram compression schemes.

2.3.4 Comparisons

For each scheme, we compute ROC curves for different gradient binning (VQ-3,5,7,9), spatial binning (DAISY-9,13,17) and quantization parameters. For a fair comparison at the same bitrate, we consider the Equal Error Rate (EER) point on the different ROC curves. The EER point is defined as the point on the ROC curve where the miss rate ($1 -$ correct match rate) and the incorrect match rate are equal. For each scheme, we compute the convex hull over the parametric space, and plot the bitrate-EER trade-off: the lower the curve, the better the performance of the descriptor.

We observe in Fig. 14(a) that Lloyd ECVQ performs best, as expected. Next, we observe that both Huffman coding and type coding schemes come close to the bound provided by Lloyd ECVQ. The type coding scheme outperforms the Huffman coding scheme at high bitrates. With type coding, we are able to match the performance of 1024-bit SIFT with about 60 bits.

In summary, we proposed two low-complexity quantization and compression schemes that come close to achieving the bound of optimal ECVQ. For each $m-$bin distribution, Huffman coding is $O(m \log m)$ in complexity, while Type Coding is $O(m)$. Both schemes do not require storage of codebooks on the mobile device, unlike ECVQ.

Finally, for reducing both speed and memory consumption, we would like to operate on descriptors in their compressed representation. We refer to this as compressed domain matching. Doing so means that the descriptor need not be decompressed during comparisons.

2.4 Compressed Domain Matching

As shown in Section 2.3, we can represent the index of the quantized distribution with fixed length codes when n is sufficiently small. To enable compressed domain matching, we pre-compute and store the distances between the different compressed distributions. This allows us to efficiently compute distances between descriptors by using indices as look-ups into a distance table. Since the distance computation only involves performing table look-ups, more effective histogram comparison measures like KL divergence and Earth Mover's Distance (EMD) can be used with no additional computational complexity. Figure 13 illustrates compressed domain matching for the VQ-5 bin configuration and quantization with Huffman trees.

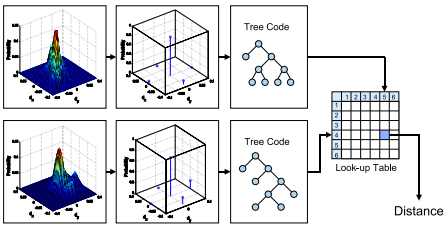


Fig. 13 Block diagram of compressed domain matching. The gradient histogram is first quantized, and mapped to an index. The indices are used to look-up the distance in a precomputed table. This figure illustrates compressed domain matching with Huffman tree quantization.

3 Experimental Results

In this section, we present a comprehensive survey of several low bitrate schemes proposed in the literature, and compare them in a common framework. First, we present feature-level ROC performance in Section 3.1, followed by image retrieval experiments in Section 3.2.

3.1 Feature Level Experiments

Here, we demonstrate that CHoG outperforms several other recent compression schemes over a wide range of bitrates. To make a fair comparison, we compare the Equal Error Rate (EER) for various schemes at the same bit rate. Fig. 14(b) compares CHoG against SIFT compression schemes proposed in the literature. Fig. 14(c) compares CHoG against other low bitrate descriptors. Here, we describe each scheme briefly, with a short discussion of its merits and drawbacks. For a more detailed description, readers are referred to (Chandrasekhar et al., 2009b, 2010b). Table 1 also summarizes the key results for the different schemes.

SIFT Compression. SIFT compression schemes can be broadly classified into three categories: hashing, transform coding and vector quantization.

- *Hashing.* We consider three hashing schemes: Locality Sensitive Hashing (LSH) (Yeo et al., 2008), Similarity Sensitive Coding (SSC) (Shakhnarovich and Darrell, 2005) and Spectral Hashing (SH). (Weiss et al., 2008) For LSH, the number of bits required to match the performance of SIFT is close to the size of the 1024-bit SIFT descriptor itself. While SSC and SH perform better than LSH at low bitrates, the performance degrades at higher bitrates due to overtraining. We note in Fig. 14(b), that there is a significant gap in performance between SIFT hashing schemes and CHoG. Hashing schemes provide the advantage of being able to compare descriptors using Hamming distances. However, note that one of the fastest techniques for computing Hamming distances is using look-up tables, a benefit that the CHoG descriptor also provides.
- *Transform Coding.* We propose transform coding of SIFT descriptors in (Chandrasekhar et al., 2009a, 2010b). In (Chandrasekhar et al., 2009a), we observe that PCA does not work well for SIFT descriptors due to its highly non-Gaussian statistics. We explore a transform based on Independent Component Analysis (ICA) in (Chandrasekhar et al., 2010b), which performs better than conventional PCA. With ICA, we can match the performance of SIFT at 160 bits.
- *Vector Quantization.* Since the SIFT descriptor is high dimensional, Jegou et al. (Jegou et al., 2010) propose decomposing the SIFT descriptor directly into smaller blocks and performs VQ on each block. The codebook index of each block is stored with fixed-length codes. The Product Quantizer (PQ) works best among all the SIFT compression schemes. We note in Fig. 14(b) that the PQ matches the performance of SIFT at 160 bits (the same bitrate is also reported in (Jegou et al., 2010)). At 160 bits, the SIFT descriptor is divided into 16 blocks, with 10 bits for each block. The size of the codebook for each block is 10^3, making it three orders of magnitude more complex than the CHoG descriptor as reported in Table 1. Further, there is still a significant gap in performance from CHoG at that bitrate. Another scheme uses a Tree Structured Vector Quantizer (TSVQ) with a million nodes. At 20 bits/descriptor, the error rate of this scheme is very high compared to other schemes. VQ based schemes require storage of codebooks, which might not be feasible on memory-limited mobile devices.

SURF Compression. We explore compression of SURF descriptors in (Chandrasekhar et al., 2009a). Transform Coding of SURF performs the best at low bitrates. The compression pipeline first applies a Karhunen-Lòeve Transform (KLT) transform (or PCA) to decorrelate the different dimensions of the feature descriptor. This is followed equal step size quantization of each dimension, and entropy coding.

Patch Compression. One simple approach to reduce bit rate is to use image compression techniques to compress canonical patches extracted from interest points. We compress 32×32 pixel patches with DA-PBT (Direction Adaptive Partition Block Transform), which is shown to perform better than JPEG (Makar et al., 2009). We compute a 128-dimensional 1024-bit SIFT descriptor

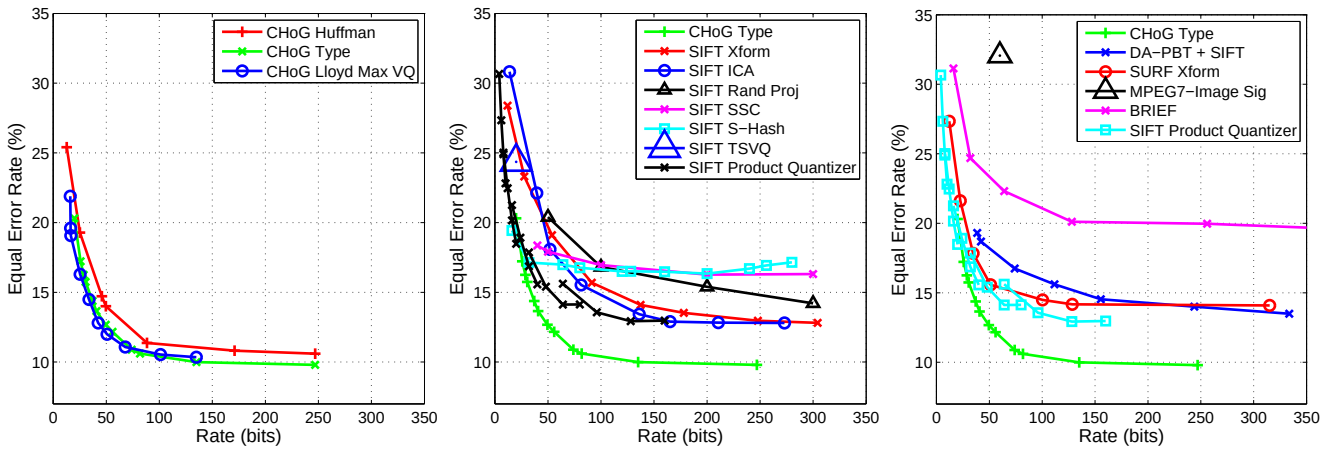


Fig. 14 Comparison of EER versus bit-rate for all compression schemes. Better performance is indicated by a lower EER. CHoG-Huffman and CHoG-Type perform close to optimal CHoG-ECVQ. CHoG outperforms all SIFT compression schemes, SURF compression schemes, MPEG-7 image signatures and patch compression over a wide range of bitrates.

on the reconstructed patch. CHoG outperforms patch compression across all bitrates.

MPEG-7 Image Signature. As part of the MPEG-7 standard, Brasnett and Bober (Brasnett and Bober, 2007) propose a 60-bit signature for patches extracted around Difference-of-Gaussian (DoG) interest points and Harris corners. The proposed method uses the Trace transform to compute a 1D representation of the image, from which a binary string is extracted using a Fourier transform. We observe in (Chandrasekhar et al., 2010a) that the descriptor is robust to simple image modifications like scaling, rotation, cropping and compression, but is not robust to changes in perspective and other photometric distortions present in the *Liberty* data sets. At 60 bits, there is a significant gap in performance between MPEG-7 image signatures and CHoG.

BRIEF. The BRIEF descriptor was proposed by Calonder et al. (Calonder et al., 2010) in their recent work. Each bit of the descriptor is computed by considering signs of simple intensity difference tests between pairs of points sampled from the patch. As recommended by the authors, the sampling points are generated from an isotropic Gaussian distribution with $\sigma^2 = S^2/25$, where $S = 64$ is the size of the patch. Simple intensity difference based descriptors do not provide the robustness of Histogram-of-Gradient descriptors, and we note that there is a significant gap in performance between BRIEF and other schemes.

Table 1 summarizes the key results from Figure 14. We note that CHoG provides the key benefits required for mobile applications: it is highly discriminative at low bitrates (matches SIFT at 60 bits), it has low complexity (linear in dimensionality of the descriptor), it requires no training and supports compressed domain matching. Next, we discuss the performance of CHoG in a practical mobile visual search application.

Scheme	# of bits	Training	Complexity	CDM
SIFT-LSH	1000	–	$O(Nd)$	√
SIFT-SSC	-	√	$O(Nd)$	√
SIFT SH	-	√	$O(Nd)$	√
SIFT-PCA	200	√	$O(d^2)$	–
SIFT-ICA	160	√	$O(d^2)$	–
SIFT-PQ	160	√	$O(Cd)$	√
SIFT-TSVQ	-	√	$O(BDd)$	√
SURF-PCA	-	√	$O(Ed)$	–
BRIEF	-	–	$O(N)$	√
CHoG	60	–	$O(d)$	√

Table 1 Results for different compression schemes. "Number of bits" column refers to the number of bits required to match the performance of 1024-bit SIFT. "Training" refers to whether or not the compression scheme requires training. "Complexity" refers to the number of operations required to compress each descriptor. "CDM" is Compressed Domain Matching. N is the number of hash-bits for the hashing schemes including BRIEF. $d = 128$ for SIFT schemes, $d = 64$ for SURF, $d = 63$ for CHoG. C = size of codebook for PQ scheme. B = breadth of TSVQ. D = depth of TSVQ.

3.2 Retrieval Experiments

In this section, we show how the low bit-rate CHoG descriptors enable novel, efficient mobile visual search applications. For such applications, one approach is to transmit the JPEG compressed query image over the network. An alternate approach is to extract feature descriptors on the mobile device and transmit them over

Fig. 15 Example image pairs from the dataset. A clean database picture (*top*) is matched against a real-world picture (*bottom*) with various distortions.

the network as illustrated in Figure 1. Feature extraction can be carried out quickly (< 1 second) on current generation phones making this approach feasible (Girod et al., 2010; Tsai et al., 2010). In this section, we study the bitrate trade-offs for the two approaches.

For evaluation, we use 3 data-sets from the literature.

- *University of Kentucky (UKY)* The UKY dataset has 10200 images of CDs, flowers, household objects, keyboards, etc (Nistér and Stewénius, 2006). There are 4 images of each object. We randomly select a set of 1000 images as query images of resolution 640×480 pixels.
- *Zurich Building Database (ZuBuD)* The ZuBuD database has 1005 images of 201 buildings in Zurich (Shao et al., 2003). There are 5 views of each building. The data set contains 115 query images of resolution 640×480 pixels.
- *Stanford Product Search (SPS)* The Stanford Product Search System is a low latency mobile product search system (Tsai et al., 2010; Girod et al., 2010). The database consists of one million CD/DVD/book cover images. The query data set contains 1000 images, of 500 × 500 pixels resolution, some illustrated in Fig. 15.

We briefly describe the retrieval pipeline for CHoG descriptors which resembles the state-of-the-art proposed in (Nistér and Stewénius, 2006; Philbin et al., 2008). We train a vocabulary tree (Nistér and Stewénius, 2006) with depth 6 and branch factor 10, resulting in a tree with 10^6 leaf nodes. For CHoG, we use symmetric KL divergence as the distance in the clustering algorithm as KL distance performs better than L_2 norm for comparing CHoG descriptors. Since symmetric KL is a Bregman divergence (Banerjee et al., 2004), it can be incorporated directly into the k-means clustering framework. For retrieval, we use the standard Term Frequency-Inverse Document Frequency (TF-IDF) scheme (Nistér and Stewénius, 2006) that represents query and database images as sparse vectors of visual word occurences, and compute a similarity between each query and database vector. We use geometric constraints to rerank the list of top 500 images (Jegou et al., 2008). The top 50 query images are subject to pairwise matching with a RAndom SAmple Consensus (RANSAC) affine consistency check. The parameters chosen enable < 1 second server-latency, critical for mobile visual search applications.

It is relatively easy to achieve high precision (low false positives) for visual search applications. By requiring a minimum number of feature matches after RANSAC geometric verification step, we obtain neglibly low false positive rates. We define Recall as the percentage of query images correctly retrieved from our pipeline. We wish to study the Recall (at close to 100% precision) vs. query size trade-offs - a high recall for small query sizes is desirable.

We compare three different schemes: (a) Transmitting JPEG compressed images, (b) Transmitting uncompressed SIFT descriptors and (c) Transmitting CHoG descriptors. Figure 16 shows the performance of the three schemes for the different data sets. For Scheme (a), we transmit a grey-scale JPEG compressed image accross the network. The bitrate is varied by changing the quality of JPEG compression. Feature extraction and matching are carried out on the JPEG compressed image on the server. We observe that the performance of the scheme deteriorates rapidly at low bitrates. At low bitrates, interest point detection fails due to blocking artifacts introduced by JPEG image compression. For Schemes (b) and (c), we extract descriptors on the mobile device and transmit them over the network. The bitrate is varied by varying the number of descriptors from 50 to 700. We pick the features with the highest Hessian response (Lowe, 2004) for a given feature budget. We observe that transmitting 1024-bit SIFT descriptors is almost always more expensive than transmitting the entire JPEG compressed image. For Scheme (c), we use a low bit-rate Type coded CHoG descriptor. We use spatial bin configuration DAISY-9, gradient bin configuration VQ-7 and type coding parameter $n = 7$, which generates a ∼70-bit descriptor. We achieve a peak recall of 96%, 94% and 75% for the SPS, ZuBuD and UKY data sets respectively. In each case, we get over an order of magnitude data reduction with CHoG descriptors, compared to JPEG compressed images or SIFT descriptors. E.g., for the SPS data set, with CHoG, we reduce data by 16× compared to SIFT, and 10× compared to JPEG compressed images.

Finally, we compare transmission times for typical cellular uplink speeds in Table 2 for the different schemes. Here, we consider the 96% highest recall point where 4 KB of CHoG data are transmitted for the SPS data set. For a slow 20 kbps link, we note that the differ-

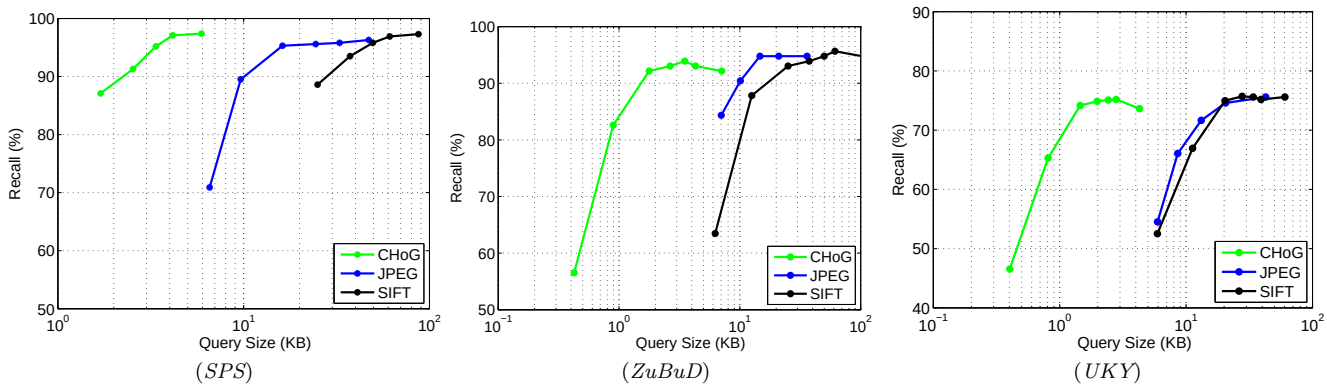


Fig. 16 Recall vs. Query Size for the Stanford Product Search (SPS), Zurich Building (ZuBuD) and University of Kentucky (UKY) data sets. High recall at low bitrates is desirable. Note that the retrieval performance of CHoG is similar to SIFT and JPEG compression schemes, while providing an order of magnitude reduction in data.

Scheme	Upload Time (s) (20 kbps link)	Upload Time (s) (60 kbps link)
JPEG+SIFT	20.0	6.7
SIFT	32.0	10.7
CHoG	1.6	0.5

Table 2 Transmission times for different schemes at varying network uplink speeds.

ence in latency between CHoG and the other schemes is about 20 seconds. We conclude that transmitting CHoG descriptors reduces query latency significantly for mobile visual search applications.

4 Conclusion

We have proposed a novel low bitrate CHoG descriptor in this work. The CHoG descriptor is highly discriminative at low bitrates, is low in complexity, and can be matched in the compressed domain, making it ideal for mobile applications. Compression of probability distributions is one of the key ingredients of the problem. To this end, we study quantization and compression of probability distributions, and propose two low complexity schemes: Huffman coding, and type coding, which perform close to optimal Lloyd Max Entropy Constrained Vector Quantization. We perform a comprehensive survey of several low bit-rate schemes and show that CHoG outperforms existing schemes at lower or equivalent bit rates. We implement the CHoG descriptor in a mobile image retrieval system, and show that CHoG feature data are an order of magnitude smaller than compressed JPEG images or SIFT feature data.

Acknowledgements We would like to thank Jana Košecka (George Mason University), Ramakrishna Vedantham, Natasha Gelfand, Wei-Chao Chen, Kari Pulli (Nokia Research Center, Palo Alto), Ngai-Man Cheung and Mina Makar (Stanford University) for their valuable input.

References

Amazon (2007) SnapTell. `http://www.snaptell.com`

Banerjee A, Merugu S, Dhillon I, Ghosh J (2004) Clustering with Bregman divergences. In: Journal of Machine Learning Research, pp 234–245

Bay H, Tuytelaars T, Gool LV (2006) SURF: Speeded Up Robust Features. In: Proc. of European Conference on Computer Vision (ECCV), Graz, Austria

Bay H, Ess A, Tuytelaars T, Gool LV (2008) Speeded-up robust feature. Computer Vision and Image Understanding 110(3):346–359, DOI http://dx.doi.org/10.1016/j.cviu.2007.09.014

Brasnett P, Bober M (2007) Robust visual identifier using the trace transform. In: Proc. of IET Visual Information Engineering Conference (VIE), London, UK

Calonder M, Lepetit V, Fua P (2010) Brief: Binary robust independent elementary features. In: Proc. of European Conference on Computer Vision (ECCV), Crete, Greece

Chandrasekhar V, Takacs G, Chen DM, Tsai SS, Girod B (2009a) Transform coding of feature descriptors. In: Proc. of Visual Communications and Image Processing Conference (VCIP), San Jose, California

Chandrasekhar V, Takacs G, Chen DM, Tsai SS, Grzeszczuk R, Girod B (2009b) 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

Chandrasekhar V, Chen DM, Lin A, Takacs G, Tsai SS, Cheung NM, Reznik Y, Grzeszczuk R, Girod B (2010a) Comparison of Local Feature Descriptors for Mobile Visual Search. In: Proc. of IEEE International Conference on Image Processing (ICIP), Hong Kong

Chandrasekhar V, Makar M, Takacs G, Chen D, Tsai SS, Cheung NM, Grzeszczuk R, Reznik Y, Girod B (2010b) Survey of SIFT Compression Schemes. In: Proc. of International Mobile Multimedia Workshop (IMMW), IEEE International Conference on Pattern Recognition (ICPR), Istanbul, Turkey

Chandrasekhar V, Reznik Y, Takacs G, Chen DM, Tsai SS, Grzeszczuk R, Girod B (2010c) Study of Quantization Schemes for Low Bitrate CHoG descriptors. In: Proc. of IEEE International Workshop on Mobile Vision (IWMV), San Francisco, California

Chou PA, Lookabaugh T, RMGray (1989) Entropy constrained vector quantization. IEEE Transactions on Acoustics, Speech and Signal Processing 37(1)

Conway JH, Sloane NJA (1982) Fast quantizing and decoding algorithms for lattice quantizers and codes IT-28(2):227–232

Cover TM, Thomas JA (2006) Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing). Wiley-Interscience

Dalal N, Triggs B (2005) Histograms of Oriented Gradients for Human Detection. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), San Diego, CA

Erol B, Antúnez E, Hull J (2008) Hotpaper: multimedia interaction with paper using mobile phones. In: Proc. of the 16th ACM Multimedia Conference, New York, NY, USA

Freeman WT, Roth M (1994) Orientation histograms for hand gesture recognition. In: Proc. of International Workshop on Automatic Face and Gesture Recognition, pp 296–301

Gagie T (2006) Compressing Probability Distributions. Information Processing Letters 97(4):133–137, DOI http://dx.doi.org/10.1016/j.ipl.2005.10.006

Girod B, Chandrasekhar V, Chen DM, Cheung NM, Grzeszczuk R, Reznik Y, Takacs G, Tsai SS, Vedantham R (2010) Mobile Visual Search. In: IEEE Signal Processing Magazine, Special Issue on Mobile Media Search, under review

Google (2009) Google Goggles. `http://www.google.com/mobile/goggles/`

Graham J, Hull JJ (2008) Icandy: a tangible user interface for itunes. In: Proc. of CHI '08: Extended abstracts on human factors in computing systems, Florence, Italy

Hua G, Brown M, Winder S (2007) Discriminant Embedding for Local Image Descriptors. In: Proc. of International Conference on Computer Vision (ICCV), Rio de Janeiro, Brazil

Hull JJ, Erol B, Graham J, Ke Q, Kishi H, Moraleda J, Olst DGV (2007) Paper-based augmented reality. In: Proc. of the 17th International Conference on Artificial Reality and Telexistence (ICAT), Washington, DC, USA

Jegou H, Douze M, Schmid C (2008) Hamming embedding and weak geometric consistency for large scale image search. In: Proc. of European Conference on Computer Vision (ECCV), Berlin, Heidelberg

Jegou H, Douze M, Schmid C (2010) Product Quantization for Nearest Neighbor Search. Accepted to IEEE Transactions on Pattern Analysis and Machine Intelligence

Johnson M (2010) Generalized descriptor compression for storage and matching. In: Proc. of British Machine Vision Conference (BMVC)

Ke Y, Sukthankar R (2004) PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. In: Proc. of Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, vol 02, pp 506–513

Kooaba (2007) Kooaba. `http://www.kooaba.com`

Kullback S (1987) The Kullback-Leibler Distance. The American Statistician 41:340–341

Lowe D (1999) Object Recognition from Local Scale-Invariant Features. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Los Alamitos, CA

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

Makar M, Chang C, Chen DM, Tsai SS, Girod B (2009) Compression of Image Patches for Local Feature Extraction. In: Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Taipei, Taiwan

Mikolajczyk K, Schmid C (2005) Performance evaluation of local descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(10):1615–1630, DOI http://dx.doi.org/10.1109/TPAMI.2005.188

Mikolajczyk K, Tuytelaars T, Schmid C, Zisserman A, Matas J, Schaffalitzky F, Kadir T, Gool LV (2005) A Comparison of Affine Region Detectors. International Journal on Computer Vision 65(1-2):43–72, DOI http://dx.doi.org/10.1007/s11263-005-3848-x

Nistér D, Stewénius H (2006) Scalable recognition with a vocabulary tree. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), New York, USA

Nokia (2006) Nokia Point and Find. `http://www.pointandfind.nokia.com`

Philbin J, Chum O, Isard M, Sivic J, Zisserman A (2008) Lost in quantization - improving particular object retrieval in large scale image databases. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Anchorage, Alaska

Rebollo-Monedero D (2007) Quantization and Transforms for Distributed Source Coding. PhD thesis, Deparment of Electrical Engineering, Stanford University

Reznik Y, Chandrasekhar V, Takacs G, Chen DM, Tsai SS, Grzeszczuk R, Girod B (2010) Fast Quantization and Matching of Histogram-based Image Features. In: Proc. of SPIE Workshop on Applications of Digital Image Processing (ADIP), San Diego, California

Rubner Y, Tomasi C, Guibas LJ (2000) The Earth Mover's Distance as a Metric for Image Retrieval. International Journal on Computer Vision 40(2):99–121, DOI http://dx.doi.org/10.1023/A:1026543900054

Shakhnarovich G, Darrell T (2005) Learning Task-Specific Similarity. Thesis

Shao H, Svoboda T, Gool LV (2003) Zubud-Zürich buildings database for image based recognition. Tech. Rep. 260, ETH Zürich

Sommerville DMY (1958) An Introduction to the Geometry of n Dimentions. Dover, New York

Takacs G, Chandrasekhar V, Gelfand N, Xiong Y, Chen W, Bismpigiannis T, Grzeszczuk R, Pulli K, Girod B (2008) 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

Tola E, Lepetit V, Fua P (2008) A fast local descriptor for dense matching. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition, pp 1–8, DOI 10.1109/CVPR.2008.4587673

Torralba A, Fergus R, Weiss Y (2008) Small Codes and Large Image Databases for Recognition. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Anchorage, Alaska

Tsai SS, Chen DM, Chandrasekhar V, Takacs G, Cheung NM, Vedantham R, Grzeszczuk R, Girod B (2010) Mobile Product Recognition. In: Proc. of ACM Multimedia (ACM MM), Florence, Italy

Weiss Y, Torralba A, Fergus R (2008) Spectral Hashing. In: Proc. of Neural Information Processing Systems (NIPS), Vancouver, BC, Canada

Winder S, Brown M (2007) Learning Local Image Descriptors. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Minneapolis, Minnesota, pp 1–8, DOI 10.1109/CVPR.2007.382971

Winder S, Hua G, Brown M (2009) Picking the best daisy. In: Proc. of Computer Vision and Pattern Recognition (CVPR), Miami, Florida

Yeo C, Ahammad P, Ramchandran K (2008) Rate-efficient visual correspondences using random projections. In: Proc. of IEEE International Conference on Image Processing (ICIP), San Diego, California