CHoG: Compressed Histogram of Gradients
A Low Bit-Rate Feature Descriptor

Vijay Chandrasekhar[†] Gabriel Takacs[†] David Chen[†] Sam Tsai[†] Radek Grzeszczuk[‡] Bernd Girod[†]

[†] Stanford University
Information Systems Lab
{vijayc,gtakacs,dmchen,sstsai,bgirod}@stanford.edu

[‡] Nokia Research Center
Palo Alto, CA
radek.grzeszczuk@nokia.com

Abstract

Establishing visual correspondences is an essential component of many computer vision problems, and is often done with robust, local feature-descriptors. Transmission and storage of these descriptors are of critical importance in the context of mobile distributed camera networks and large indexing problems. We propose a framework for computing low bit-rate feature descriptors with a 20× reduction in bit rate. The framework is low complexity and has significant speed-up in the matching stage. We represent gradient histograms as tree structures which can be efficiently compressed. We show how to efficently compute distances between descriptors in their compressed representation eliminating the need for decoding. We perform a comprehensive performance comparison with SIFT, SURF, and other low bit-rate descriptors and show that our proposed CHoG descriptor outperforms existing schemes.

1. Introduction

Local image features have become pervasive in the areas of computer vision and image retrieval. These features are increasingly finding applications in real-time object recognition [1], 3D reconstruction [2], panorama stitching [3], robotic mapping [4], and video tracking [5]. Depending on the application, either transmission or storage issues (or both) can limit the speed of computation or the size of databases. In the context of mobile devices (camera phones) or distributed camera networks, communication and power costs are significant for transmitting information between nodes. Feature compression is hence vital for reduction in storage, latency and transmission.

Server-side Storage: Image retrieval applications need query images to be matched against databases of millions of features stored at application servers. Feature compression can yield significant savings in storage space,

Application Latency: When data are sent over a network, the system latency can be reduced by sending fewer bits resulting from compression of image features.

Data Transmission: For mobile applications, bandwidth is a limiting factor. Feature compression can reduce the amount of data transmitted over wireless channels and backhaul links in a mobile network.

Motivated by these demands, we propose a new framework for computing low bit-rate feature descriptors.

1.1. Prior Work

Research on robust local descriptors continues to be a very active area of computer vision. Of the many proposed descriptors, SIFT [6] is probably the most commonly used. Other popular descriptors include GLOH by Mikolajczyk and Schmid [7], and SURF by Bay *et al.* [8]. The review paper by Mikolajczyk *et al.* [9] provides a comprehensive analysis of several descriptors. Winder and Brown [10] also investigate various published descriptors, and propose a framework for optimizing parameters in the descriptor computation process.

Low-bit-rate descriptors are of increasing interest in the vision community. Often, feature data are reduced by decreasing the dimensionality of descriptors. Ke and Sukthankar [11] investigate dimensionality reduction via Principle Component Analysis. Hua *et al.* [12] propose a scheme that uses Linear Discriminant Analysis. Takacs *et al.* [1] quantize and entropy code SURF feature descriptors for reducing their bit rate. Chandrasekhar *et al.* [13] propose a general framework for transform coding image features. Chuohao *et al.* [14] reduce the bit rate of descriptors by using random projections on SIFT descriptors to build binary hashes. Descriptors are then compared using their binary hashes. Shakhnarovich, in his thesis [15], uses a machine learning technique called Similarity Sensitive Coding to train binary codes on image patches.

1.2. Contributions

We first show how to capture gradient statistics from canonical patches in a histogram. In contrast to prior work, we explicity exploit the underlying gradient statistics that

978-1-4244-3991-1/09/$25.00 ©2009 IEEE

arise from canonical patches around interest points. Retaining the statistics as a histogram enables us to use more effective distance measures like Kullback-Leibler (KL) divergence and Earth Mover's Distance (EMD) for comparing descriptors. We then use tree coding techniques to quantize and compress these histograms into low bit-rate feature descriptors. We show how these Compressed Histogram of Gradients (CHoG) descriptors can be compared in the compressed domain. This reduces the computational complexity by eliminating decompression from distance computations. We show that exact nearest-neighbor searching for CHoG is 10× faster than for SIFT. Finally, we compare CHoG to several low-bit-rate descriptors and show that CHoG outperforms other schemes at equivalent bit rates.

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 for comparing histograms. In this section we choose the parameters of our Uncompressed Histogram of Gradients (UHoG) descriptor. We first describe the framework used for evaluation.

2.1. Descriptor Evaluation

For evaluating the performance of low-bit-rate descriptors, we use two data sets provided by Winder and Brown [10], *Trevi Fountain* and *Notre Dame*. Each data set has matching pairs of 64×64 pixel patches. For algorithms that require training, we use matching pairs from the *Trevi Fountain*. For testing, we randomly select 10,000 matching pairs and 10,000 non matching pairs from the *Notre Dame* set. In addition to the process used by Winder and Brown, we orient all patches in the direction of the most dominant gradient. Patch orientation ensures that the gradient statistics are similar to the statistics of canonically oriented patches obtained from interest point detectors.

We use the method proposed by Winder and Brown [10] for descriptor evaluation. We compute a distance between each pair of descriptors. From these distances, we form two histograms, one for matching pairs and one for non-matching pairs. From the two histograms we obtain a Receiver Operating Characteristic (ROC) curve which plots correct match fraction against incorrect match fraction.

The 128-dimensional SIFT descriptor [6] is considered the gold standard of descriptors [7], and hence we use it as our reference descriptor. We also compare against the 64-dimensional SURF [8] descriptor, which works well in practice. Both SIFT and SURF are quantized to 8 bits, resulting in 1024 and 512 bit descriptors, respectively.

2.2. Histogram of Gradient Based Descriptors

A number of different feature descriptors are based on the distribution of gradients within a patch of pixels. In this Section, we describe the pipeline used to compute gradient histogram descriptors. We then show the relationships between SIFT, SURF and UHoG.

As in [9], 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 cells, which gives robustness to interest point localization error. Then, some statistics of d_x and d_y are extracted seperately for each cell, forming the descriptor.

SIFT and SURF descriptors can be calculated as functions of the gradient histograms, provided that such histograms are available for each cell 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 cell. Note that the gradients within a cell may be weighted by a Gaussian window prior to descriptor computation [6, 8].

The 8 SIFT components of a cell, $\mathscr{D}_{SIFT}$, are

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

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

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

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

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

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

In contrast to SIFT and SURF, we propose coarsely quantizing the 2D gradient histogram, and capturing the histogram directly into the 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 Figure 1. We refer to this uncompressed descriptor representation as Uncompressed Histogram of Gradients (UHoG). The i^{th} UHoG descriptor is defined as $\mathscr{D}^i_{UHoG} = \left[\hat{P}^i_1, \hat{P}^i_2, \ldots, \hat{P}^i_N\right]$, where $\hat{P}^i_k$ represents the gradient

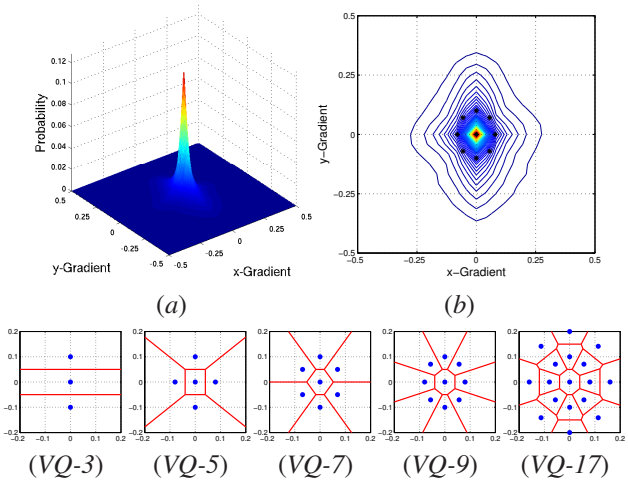


Figure 1. The joint (d_x, d_y) gradient distribution (a) over a large number of cells, and (b), its contour plot. The greater variance in y-axis results from aligning the patches along the most dominant gradient after interest point detection. The quantization bin constellations VQ-3, VQ-5, VQ-7, VQ-9 and VQ-17, and their associated veronoi cells are shown at the bottom.

histograms in cell k of descriptor i, and N is the total number of cells. Similar gradient histogram binning techniques have also been proposed by Dalal and Triggs [16], and Freeman and Roth [17]. Note that the dimensionality of UHoG is given by $N \times B$, where N is the number of cells, and B is the number of bins in the gradient histogram.

2.3. 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 Figure 1(a,b). We observe that the distribution is strongly peaked around $(0,0)$, and that the variance is higher 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 (VQ) of the gradient distribution into a small set of bin centers, S, shown in Figure 1. We call these bin configurations VQ-3, VQ-5, VQ-7, VQ-9, and VQ-17. 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 over ellipses, the eccentricity of which are chosen in accordance with the observed skew in the gradient statistics.

To evaluate the performance of each bin configuration we plot the ROC curves in Figure 2. For these curves, we use KL divergence, which we show in Section 2.5 performs

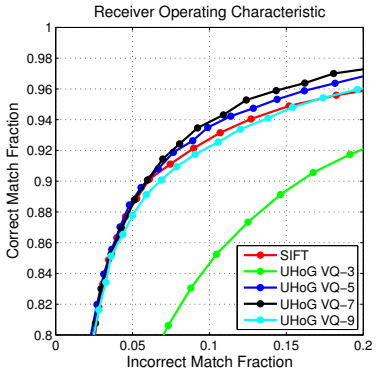


Figure 2. ROC curves for various bin configurations. GRID-16 cell configuration and KL divergence are used. The VQ-5 configuration matches the performance of SIFT.

well as the distance measure for histogram comparison. As we increase the number of bin centers, we obtain a more accurate approximation of the gradient distribution. As a result, the performance of the descriptor improves until VQ-7. After VQ-7, we begin to over-quantize the distribution leading to a less robust descriptor. We observe that the VQ-5 configuration with 5 bins suffices to match the performance of SIFT.

2.4. Cell Configuration

Since we want the smallest possible feature descriptor, we have experimented with reducing the number of cells. Fewer cells means fewer histograms and a smaller descriptor. However, it is important that we do not adversely affect the performance of the descriptor.

SIFT [6] and SURF [8] use a square 4×4 grid with 16 cells (GRID-16). Mikolajczyk *et al.*[7] propose using log-polar cell configurations (GLOH-9, GLOH-7). Here, we consider these three different cell configurations, shown in Figure 3. The performance of GLOH-7 matches that of GRID-16, and provides a dimensionality reduction of 56% compared to GRID-16. GLOH-9 performs better than GRID-16, while providing a dimensionality reduction of 44%. We therefore select GLOH-9 as the cell configuration for low bit-rate descriptors.

2.5. 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 [18], and the Earth Mover's Distance [19]. The distance between two UHoG (or CHoG) descriptors is defined as $d(\mathscr{D}^i, \mathscr{D}^j) = \sum_{k=1}^{N} d_{\text{hist}}(\hat{P}_k^i, \hat{P}_k^j)$, where N is the number of cells, d_{hist} is a distance measure between two distributions, and $\hat{P}^i$ represents the gradient distribution in a cell.

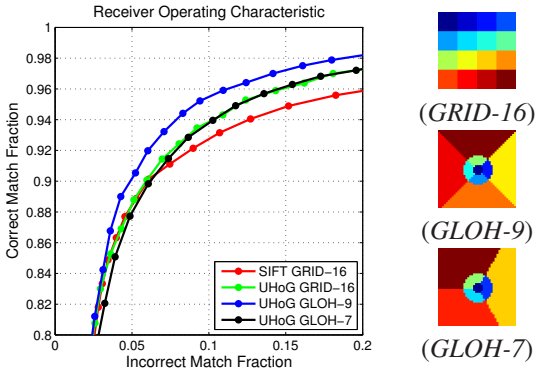


Figure 3. ROC curves for various cell configurations. VQ-5 bin configuration and KL divergence are used. The log-polar GLOH-9 cell configuration performs better than GRID-16.

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,

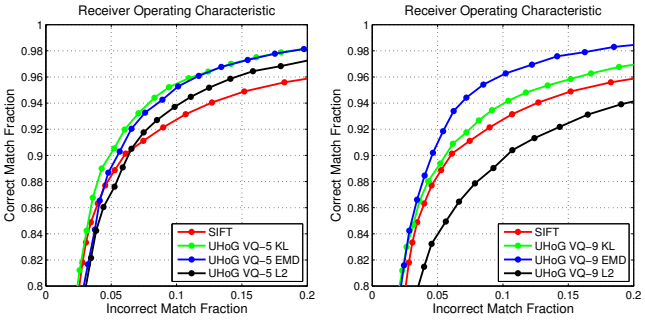
$$d_{\mathrm{KL}}(\hat{P}^i, \hat{P}^j) = \sum_{n=1}^{B} p^i_n \log \frac{p^i_n}{p^j_n} + \sum_{n=1}^{B} p^j_n \log \frac{p^j_n}{p^i_n}. \qquad (6)$$

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 Figure 1. Note that EMD is a metric and follows the triangle inequality, while KL divergence is not.

In Figure 4 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, particularly for VQ-9. These results show that gradient histograms can be compared effectively by using KL divergence. This motivates techniques to compress probability distributions which provide a bound on the distortion in KL divergence.

3. Descriptor Compression

Keeping the descriptor as a set of histograms not only allows us to exploit patch statistics and meaningful distance metrics, but also allows us to leverage work from the information theory literature on histogram compression. Our goal is to produce low bit-rate Compressed Histogram of Gradients (CHoG) descriptors while maintaining the highest possible fidelity. In this section we demonstrate that we can represent a histogram as a tree structure and thereby reduce the data while keeping an upper bound on the resulting distortion. A block diagram of the compression pipeline is shown in Figure 5.



(a) (b)

Figure 4. ROC curves for distance measures with bin configurations VQ-5 (*b*) and VQ-9 (*b*). KL and EMD consistently outperform L_2-norm.

3.1. Tree Coding

Lossy compression of probability distributions is an interesting problem that has has not received much attention in the compression literature. Gagie [20] recently proposed two algorithms for compressing probability distributions with a bound on the KL divergence between the original and compressed distributions. We refer to these two algorithms as *Gagie* and *Huffman* tree coding. For both algorithms, let $P = [p_1, p_2, ... p_n] \in \mathbb{R}^{n+}$ be the original distribution, and $Q = [q_1, q_2, ... q_n] \in \mathbb{R}^{n+}$ be the lossily quantized probability distribution defined over the same sample space.

3.1.1 Huffman Tree Coding

Given a probability distribution, one to way to compress it is to construct and store a Huffman tree built from the distribution. 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. As shown in [20], Huffman tree coding guarantees that $D(P||Q) < 1$, where $D(P||Q) = \sum_{i=1}^{n} p_i \log_2 \frac{p_i}{q_i}$.

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 need not be stored, since a Huffman tree is a strict binary tree and $\sum q_i = 1$. An example of constructing Huffman tree codes for a VQ-5 gradient histogram is shown in Figure 6.

3.1.2 Gagie Tree Coding

The Gagie tree algorithm constructs a distribution, Q, such that $D(P||Q) < \log_2(2 + 2^{3-k})$, where Q can be stored in exactly $kn - 2$ bits. Readers are referred to [20] for the proof and algorithms. Here, we present the special case of the algorithm with $k = 2$. The compression provides

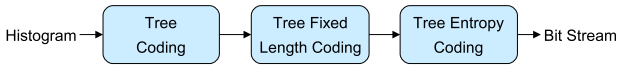


Figure 5. Block diagram for histogram compression. Distributions are quantized with Huffman or Gagie trees, which are encoded directly for large n, or with fixed-length or entropy coding for small n.

$D(P||Q) < 2$, and the distribution can be stored in exactly $2n-2$ bits as a strict ordered binary tree.

We compute a Gagie Tree as follows,

1. Compute $S = \{s_1, \ldots, s_n\}$, such that $s_i = \frac{p_i}{2} + \sum_{j=1}^{i-1} p_j$.
2. Construct codes, c_i, to be the first b_i bits of s_i, where $b_i = \lceil \log(\frac{2}{p_i}) \rceil$. The c_i from a prefix-free code.
3. From the c_i, construct an ordered, binary tree, $\mathscr{T}'_{\text{Gagie}}$. Prune internal nodes of $\mathscr{T}'_{\text{Gagie}}$ that have only one child to obtain a strict, ordered, binary tree $\mathscr{T}_{\text{Gagie}}$.
4. Represent $\mathscr{T}_{\text{Gagie}}$ succinctly in $2n-2$ bits as a sequence of balanced parentheses [21].

An example of constructing Gagie tree codes is shown in Figure 6. Gagie trees are ordered and, hence, the tree itself stores the entire distribution P. However, Huffman trees are not ordered, because symbol probabilities are sorted in the tree building process. The Huffman tree results in a lower $D(P||Q)$, but requires a higher number of bits, $(n-1)\lceil \log(n-1) \rceil$, compared to $2n-2$ bits for Gagie trees. While the tree coding schemes can be used for all n, we show how to reduce the bit rate further for small n in the next Section.

3.2. Tree Fixed Length Coding

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 possible Gagie trees is given by the Catalan number,

$$C_{n-1} = \frac{1}{n}\binom{2n-2}{n-1}, \tag{7}$$

which yields the number of strict, ordered, binary trees with n leaf nodes. The Huffman trees are a superset of the Gagie trees which can be enumerated as all unique permutations of the leaf nodes of Gagie trees. A simple example of all possible Gagie and Huffman trees with 4 leaf nodes is illustrated in Figure 7. The tree enumeration technqiue can be employed for $n \leq 14$ for Gagie trees, and $n \leq 9$ for Huffman trees. Beyond this, enumeration is not practical due to the large number of possible trees. Figure 8 shows the reduction in bit rate obtained by using fixed length coding. We next show how the bit rate can be further reduced for $n \leq 7$ for both schemes.

P	S	C	Gagie Tree	Q
0.8281	0.4141	0		0.500
0.0000	0.8281	11010		0.125
0.0312	0.8438	11011		0.125
0.0625	0.8906	1110		0.125
0.0781	0.9609	1111		0.125

P	Sort(P)	Huffman Tree	C	Q
0.8281	0.0000		1	0.5000
0.0000	0.0312		0000	0.0625
0.0312	0.0625		0001	0.0625
0.0625	0.0781		001	0.1250
0.0781	0.8281		01	0.2500

Figure 6. Example of Gagie (*top*) and Huffman (*bottom*) tree coding. Both schemes start with the same distribution, P, and result in the compressed distribution, Q. This example results in a KL divergence of 0.2945 and 0.2620 for Gagie and Huffman, respectively.

3.3. Tree Entropy Coding

We can achieve further compression by entropy coding the fixed-length indices. This is because not all trees are equally likely to occur from gradient statistics. For this compression we use an arithmetic coder. The resulting bit-rate reduction for $n \leq 7$ is illustrated in Figure 8. Entropy coding is not practical for large n as the statistics cannot be reliably estimated for such a large sample space.

3.4. Compression Results

After presenting two different methods of compressing gradient distributions we now compare their performance. For both Huffman and Gagie schemes, we pick the lowest possible bit-rate for a given n, as shown in Figure 8.

Figure 9 shows the results of compressing descriptors using Gagie (*a*) and Huffman (*b*). For both schemes, the bit rate increases as the number of bins increases, and so does the performance of the descriptor. The Gagie scheme can achieve close to the performance of SIFT using the VQ-9 configuration with 99 bits. The Huffman scheme can match

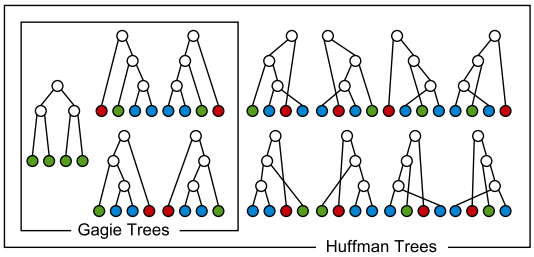
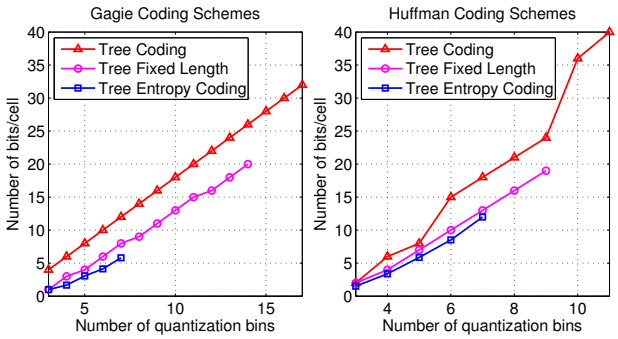
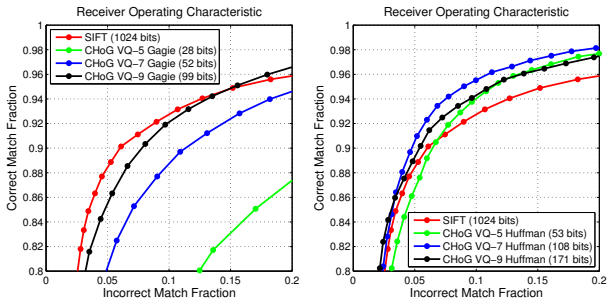


Figure 7. All possible unique, strict, binary trees with 4 leaves. Huffman trees are a superset of Gagie trees which can be constructed as unique permutations of the leaves of Gagie trees. The leaves are color coded by depth.



(a) (b)

Figure 8. Bit-rate per cell for Huffman and Gagie histograms using different schemes. We can obtain 2-4× compression compared to the Tree Coding representation. Note the ranges of n in which Tree Fixed Length and Tree Entropy Coding can be used.



(a) (b)

Figure 9. ROC curves for compressing distributions with Gagie and Huffman compression schemes with GLOH-9 and KL divergence. In (*a*) and (*b*) we can match the performance of SIFT with 99 and 53 bits for Gagie and Huffman compression schemes, respectively.

the performance of SIFT using VQ-5 with 53 bits, while the VQ-7 configuration at 108 bits outperforms SIFT.

Additionally, we note that compression with Huffman tree coding affects the performance of the descriptor less than compression with Gagie trees. This results from the smaller bound on KL divergence that arises from compressing distributions with Huffman trees. Importantly, we observe that the performance of the descriptor compressed with Huffman Tree compression is close to the performance of the uncompressed descriptor.

Finally, for a fair comparison of Huffman and Gagie Tree compression schemes at the same bit rate, we consider the Equal Error Rate (EER) point on the different ROC curves for each scheme. 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. As shown in Figure 10, the Huffman compression scheme provides a lower EER for a given bit-rate than Gagie compression.

4. Descriptor Matching

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. Additionally, computational complexity is reduced by eliminating compression and decompression from distance computations. In this Section, we show how to do 10× faster nearest neighbor search with CHoG features in their compressed representation.

4.1. Compressed Domain Matching

As shown in Section 3.2, we can represent the index of Huffman and Gagie trees with fixed length codes when n is sufficiently small. To allow for 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 tree 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 EMD can be used with no additional computational complexity. Figure 11 illustrates compressed domain matching for the VQ-5 bin configuration.

4.2. Accelerated Search Strategies

There are many important uses of feature descriptors that require nearest-neighbor searches. Some examples include vocabulary trees [22] and nearest neighbor matching of features [23]. One of the most commonly used data structures is an approximate kd-tree (ANN) [24]. However, since we wish to use arbitrary metrics as well as compressed domain matching, kd-trees cannot be used with CHoG descriptors. To enable $O(\log n)$ search, we use distance metrics such as EMD or L_2-norm, along with a metric ball tree. The tree exploits the Triangle Inequality while forming a binary tree using only distances between descriptors. Metric-trees can easily be extended for approximate searches with a $(1+\varepsilon)$ guarantee in the same way as [24].

Table 1 shows the timing results for nearest-neighbor querying with both SIFT and CHoG. For the experiment, 10^3 features descriptors were queried into a database of 10^6 descriptors. The descriptors for both SIFT and CHoG were computed from the same set of patches. For CHoG, we use a 45-dimensional descriptor resulting from GLOH-9, VQ-5 and Huffman Tree compression. Exact nearest neighbor searching is ∼10× faster for CHoG than SIFT. Furthermore, CHoG is 2× faster than SIFT with ANN ($\varepsilon = 1$), which incurs an error rate of 0.30%. The speed-up comes from lower dimensionality, and the use of look-up tables for fast distance computation. Additionally, for matching against a small number of feature descriptors (∼1000), optimized brute-force search is fast, and the KL divergence can be used for better ROC performance.

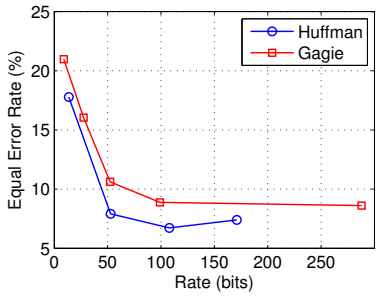


Figure 10. Comparing Huffman and Gagie coding schemes. The Huffman scheme provides a lower EER for a given bit-rate than the Gagie scheme.

5. Comparisons

We now demonstrate that CHoG outperforms several other recent compression schemes. To make a fair comparison, we compare the Equal Error Rate (EER) for various schemes at the same bit rate. Figure 12 compares CHoG with Huffman or Gagie compression against the other schemes. Each scheme is briefly described in the following Sections.

5.1. Patch Compression

One simple approach to reduce bit rate is to use image compression techniques to compress canonical patches extracted from interest points. SIFT descriptors can then be extracted from these compressed patches. We compress 32×32 pixel patches with JPEG [25], and then compute a 128-dimensional 1024-bit SIFT descriptor on the reconstructed patch. As seen in Figure 12, JPEG requires 295 bits to compete with the performance of CHoG-Huffman at 53 bits.

5.2. Random Projections

Chuohao *et al.* [14] propose the use of quantized random projections to build binary hashes from SIFT descriptors. Hamming distance between hashes is used as the distance measure. The Random Projection scheme requires more than 500 bits to compete with the performance of CHoG-Huffman at 53 bits. A practical disadvantage of random hashing is that the computational complexity increases with the number of hash bits.

	Time (sec)	Error (%)
CHoG Metric Tree	27.96	0.00
SIFT *k*d-tree Exact NN	371.80	0.00
SIFT *k*d-tree Approx NN	47.01	0.30

Table 1. Nearest-neighbor run-times on a 2 GHz Intel Xeon processor compared for SIFT and CHoG with L_2-norm. There are 10^3 queries into a database of size 10^6.

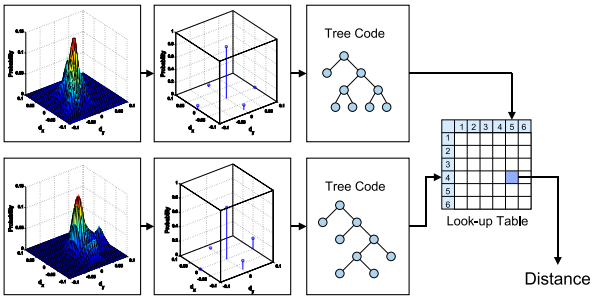


Figure 11. Block diagram of compressed domain matching. The gradient histogram is first quantized, and then tree coded. The tree indices are used to look-up the distance in a precomputed table.

5.3. Boosting Similarity Sensitive Coding

Shakhnarovich [15] uses a machine learning algorithm called Boosting Similarity Sensitive Coding (Boost SSC) [26] to train binary codes that reflect patch similarity. The Boosting SSC algorithm learns an embedding of the original Euclidean space into a binary space such that the hamming distance is correlated with Euclidean distance.

We trained Boosting SSC with 10000 matching SIFT descriptor pairs and 40000 non-matching pairs. Figure 12 shows that Boosting SSC outperforms random projections at low bit rates. However, at higher bit rates, the performance of the scheme degrades due to over-training. Additionally, Boosting SSC requires an expensive training step, while CHoG does not.

5.4. Transform Coding

Transform coding of SURF and SIFT descriptors was proposed by Chandrasekhar *et al.* [13]. The compression pipeline first applies a Karhunen-Lòeve Transform (KLT) transform to decorrelate the different dimensions of the feature descriptor. This is followed equal step size quantization of each dimension. The quantized features are entropy coded with an arithmetic coder. SIFT-Transform coding requires 270 bits to match the performance of CHoG-Huffman at 53 bits. SURF-Transform coding requires 133 bits to match the performance of 53-bit CHoG-Huffman.

5.5. Tree Structured Vector Quantization

Nistér and Stewénius [22] proposed the idea of using hierarchical *k*-means to quantize SIFT descriptors, leading to a Tree Structured Vector Quantizer (TSVQ). Here, we quantize SIFT descriptors with a 10^6 node TSVQ with a branch factor of 10 and depth of 6, requiring 20 bits per descriptor. A significantly larger TSVQ is not practical due to the size of the code book. As seen in Figure 12, TSVQ compression performs poorly and does not come close to the performance of CHoG.

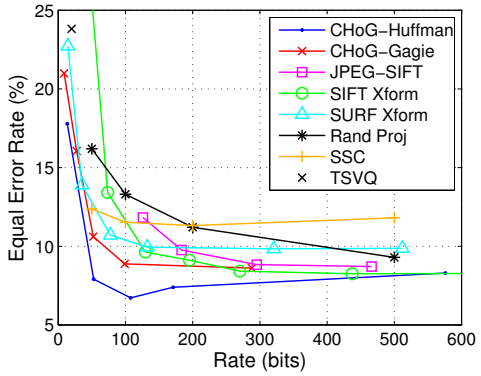


Figure 12. Comparison of EER versus bit-rate for all compression schemes. Better performance is indicated by a lower EER. CHoG-Huffman at 53 bits outperforms all the other schemes.

6. Conclusion

We have proposed a novel framework for computing low bit-rate feature descriptors with 20× reduction in bit rate, low complexity and significant speed-up in the matching stage. By quantizing and encoding gradient histograms with Huffman and Gagie trees we can create very low bit-rate descriptors. We efficently compute distances between descriptors in their compressed representation eliminating the need for decoding. We perform a comprehensive performance comparison with SIFT, SURF, and other low bit-rate descriptors and show that CHoG outperforms existing schemes at lower or equivalent bit rates. The CHoG descriptor enables many applications that are limited by bandwidth, storage, or latency. Such applications include mobile augmented reality, large-scale image retrieval, and visual sensor networks.

7. Acknowledgments

We would like to thank Professor Jana Košecka for her valuable input and guidance.

References

[1] G. Takacs, V. Chandrasekhar, N. Gelfand, Y. Xiong, W.-C. Chen, T. Bismpigiannis, R. Grzeszczuk, K. Pulli, and B. Girod, "Outdoors Augmented Reality on Mobile Phone using Loxel-Based Visual Feature Organization," *ACM International Conference on Multimedia Information Retrieval (MIR)*, 2008. 1

[2] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo Tourism: Exploring Photo Collections in 3D," in *SIGGRAPH Conference Proceedings*. New York, NY, USA: ACM Press, 2006, pp. 835–846. 1

[3] M. Brown and D. Lowe, "Automatic Panoramic Image Stitching Using Invariant Features," in *International Journal of Computer Vision*, vol. 74, no. 1, 2007, pp. 59–77. 1

[4] S. Se, D. Lowe, and J. Little, "Vision-Based Global Localization and Mapping for Mobile Robots," in *IEEE Transactions on Robotics*, vol. 21, no. 3, 2007, pp. 364–375. 1

[5] G. Takacs, V. Chandrasekhar, B. Girod, and R. Grzeszczuk, "Feature Tracking for Mobile Augmented Reality Using Video Coder Motion Vectors," in *ISMAR '07: Proceedings of the Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007. 1

[6] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004. 1, 2, 3

[7] K. Mikolajczyk and C. Schmid, "Performance Evaluation of Local Descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1615–1630, 2005. 1, 2, 3

[8] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: Speeded Up Robust Features," in *ECCV (1)*, 2006, pp. 404–417. 1, 2, 3

[9] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool, "A Comparison of Affine Region Detectors," *Int. J. Comput. Vision*, vol. 65, no. 1-2, pp. 43–72, 2005. 1, 2

[10] S. A. Winder and M. Brown, "Learning Local Image Descriptors," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, 2007, pp. 1–8. 1, 2

[11] Y. Ke and R. Sukthankar, "PCA-SIFT: A More Distinctive Representation for Local Image Descriptors," in *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 02. IEEE Computer Society, 2004, pp. 506–513. 1

[12] S. W. G. Hua, M. Brown, "Discriminant Embedding for Local Image Descriptors," in *Proc. of International Conference on Computer Vision (ICCV)*. IEEE Computer Society, 2007. 1

[13] V. Chandrasekhar, G. Takacs, D. Chen, S. Tsai, and B. Girod, "Transform Coding of Feature Descriptors," in *VCIP*, 2009. 1, 7

[14] C. Yeo, P. Ahammad, and K. Ramchandran, "Rate-Efficient Visual Correspondences Using Random Projections," 2008. 1, 7

[15] G. Shakhnarovich and T. Darrell, "Learning Task-Specific Similarity," *Thesis*, 2005. 1, 7

[16] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893 vol. 1, June 2005. 3

[17] W. T. Freeman and M. Roth, "Orientation Histograms for Hand Gesture Recognition," 1995, pp. 296–301. 3

[18] S. Kullback, "The Kullback-Leibler Distance," *The American Statistician*, vol. 41, 1987. 3

[19] Y. Rubner, C. Tomasi, and L. J. Guibas, "The Earth Mover's Distance as a Metric for Image Retrieval," *Int. J. Comput. Vision*, vol. 40, no. 2, pp. 99–121, 2000. 3

[20] T. Gagie, "Compressing Probability Distributions," *Inf. Process. Lett.*, vol. 97, no. 4, pp. 133–137, 2006. 4

[21] J. I. Munro and V. Raman, "Succinct Representation of Balanced Parentheses, Static Trees and Planar Graphs," in *FOCS '97: Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS '97)*. Washington, DC, USA: IEEE Computer Society, 1997, p. 118. 5

[22] D. Nistér and H. Stewénius, "Scalable Recognition with a Vocabulary Tree," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2006, pp. 2161–2168. 6, 7

[23] M. Brown and D. Lowe, "Unsupervised 3D Object Recognition and Reconstruction in Unordered Datasets," in *5th International Conference on 3D Imaging and Modelling (3DIM05)*, Ottawa, Canada, 2005. 6

[24] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An Optimal Algorithm for Approximate Nearest Neighbor Searching Fixed Dimensions," *Journal of the ACM*, vol. 45, no. 6, pp. 891–923, 1998. 6

[25] "ISO/IEC 10918-1 ITU-T Recommendmation T.81, Information Technology: Digital Compression and Coding of Continuous-Tone Still Images," 1992. 7

[26] G. Shakhnarovich, P. Viola, and T. Darrell, "Fast Pose Estimation with Parameter-Sensitive Hashing," in *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*. Washington, DC, USA: IEEE Computer Society, 2003, p. 750. 7