Generalized Descriptor Compression for Storage and Matching

Matthew Johnson
http://www.matthewajohnson.org/

Nokia Point and Find
35th Floor
1 Market Plaza
San Francisco, CA

Abstract

Smarter phones have made handheld computer vision a reality, but limited bandwidth, storage space and processing power prevent mobile phones from leveraging the full body of existing research. In particular, common techniques which use feature detectors and descriptors to solve problems in image matching and augmented reality cannot be used due to their space and processing requirements. We propose a general descriptor compression method which reduces descriptor size and provides fast descriptor matching without requiring decompression. By demonstrating how to apply our method to the commonly used SIFT, SURF and GLOH descriptors, we show its effectiveness in reducing size and increasing accuracy. In all cases, we reduce the size of the descriptor by an order of magnitude and achieve higher accuracy at a detection rate of 95%.

1 Introduction

With the advent of smart phones that incorporate high quality cameras, mobile computer vision has become an area of increased research. While many efforts have focused on thin phone clients which interact with a remote server that uses conventional computer vision techniques to solve image matching and recognition problems [5, 9, 18], the current generation of phones are sufficiently powerful that they can execute computer vision algorithms locally. In particular, ubiquitous augmented reality on the mobile phone becomes a feasible undertaking, as the complementary sensors (*e.g.* GPS, accelerometer) work along with the camera to provide a rich user experience. Such applications heavily utilize interest point and descriptor based algorithms for matching, registration and tracking. However, these algorithms present unique challenges in a mobile computer vision scenario:

Storage Space Memory is in short supply on mobile handsets, and thus the large memory footprints of descriptors like SIFT [12], SURF [2] and GLOH [13] make the storing of databases in memory unfeasible, and writing to and from the disk on a mobile phone is too slow for the realtime recognition required for an augmented reality application.

Bandwidth The local image descriptors computed for a reference image or object number in the thousands and each require a large number of bits to represent. Given the bandwidth available in a mobile scenario, providing a localized database from a remote server as a user moves from one geographical area to another becomes untenable.

Computation The most computationally intensive component of any image matching or registration system is descriptor database retrieval. With thousands of descriptors in the image and thousands more in the database, millions of distance computations can potentially take place.

In order to allow mobile computer vision to exploit the substantial body of computer vision research depending upon existing local image descriptor algorithms despite the challenges listed above, we propose a general and efficient method of compression for descriptors, and a technique by which those descriptors can be compared without requiring decompression. An overview of our method can be seen in Figure 1.

1.1 Prior Work

There has been a significant effort in recent years to produce robust local descriptors for a variety of imaging tasks including image matching, pose retrieval, and object recognition. The most widely used are Lowe's SIFT descriptor [12], the SURF descriptor of Bay *et al*. [2], and the GLOH descriptor proposed by Mikolajczyk and Schmid in [13]. That paper also provides an excellent review of various descriptors types and methods for their evaluation. Another more current review of descriptors was performed by Winder and Brown in [17], in which they also explore methods for optimizing the different parameters involved in the design of descriptors using machine learning techniques.

Accompanying this research into robust local descriptors has been complementary work on descriptor compression. Ke and Sukthankar reduce the dimensionality of a 39x39 patch of x and y gradients using principle components analysis to obtain their 36-dimensional PCA-SIFT descriptor[10], one of several examples where PCA is used to reduce the dimensionality of a larger descriptor to ease storage requirements (another is the GLOH descriptor of Mikolajczyk and Schmid [13]). Hua *et al*. [7] propose a learned linear discriminant system which accomplishes the same goal on standard descriptors. Takacs *et al*. [16] quantize and entropy code SURF descriptors to reduce the bitrate, and Chandrasekhar *et al*. propose a general framework in [3] for the transform coding of feature descriptors, though this scheme requires decompression during the matching phase. Yeo *et al*. in [19] use random projections on SIFT descriptors to create compact binary hashes. Another interesting approach is taken by Shaknarovich in his thesis [15], where he develops a machine learning technique called Similarity Sensitive Coding to learns efficient binary codes directly from image patches.

Most similar to our work, Chandrasekhar *et al*. propose a new descriptor in [4] called the Compressed Histogram of Gradients, which is compressed using a method which can be seen as a special instance of our generalized method. Their version of a histogram of gradients is a descriptor (they refer to it as a UHoG) which takes our canonical form without modification. The histogram is small enough that a generalized encoding scheme is not used, but rather each particular coding tree is assigned a unique identifier. In this paper, we show how the advantages of their CHoG descriptor can be gained by any descriptor, thus enabling a large body of existing research to be leveraged in mobile computer vision scenarios.

1.2 Contributions

We first propose a *canonical descriptor form*, and then demonstrate a technique by which a descriptor in this form can be significantly reduced in size using tree coding. We then describe the manner in which the distances between these compressed descriptors can be

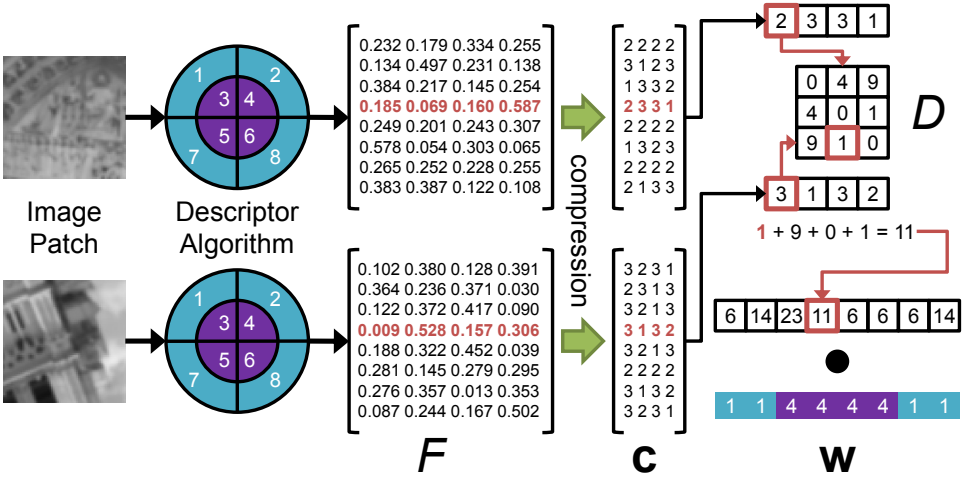

Figure 1: **Method Overview** In this figure, we can see an overview of the entire compression and matching method. A descriptor algorithm (in this case, a GLOH-like scheme) is used to compute a matrix F for two image patches, in which each row is a normalized histogram which corresponds to a cell in a mask over the patch. F is then compressed to form the vector $\mathbf{c}$, where each index represents a row from F. Each element in this vector is then used to compute per-histogram distances by way of a distance matrix D. The dot product of this vector and a weight vector $\mathbf{w}$ produce the final distance. F, D and $\mathbf{w}$ constitute our proposed *canonical form* of a descriptor algorithm.

computed without requiring decompression, as is the case in other universal techniques. Finally, to demonstrate the efficacy of the technique, we examine three case studies for SIFT, SURF, and GLOH descriptors. In each study, we describe how to convert the descriptor algorithm into the canonical form and demonstrate how descriptors compressed from this form using our method perform better than their uncompressed versions while requiring far fewer bits to encode.

2 Descriptor Compression

We propose a *canonical descriptor form* uniquely suited for compression which captures the structure shared by most local image descriptors. This form consists of three parts, as seen in Figure 1:

1. An $M \times N$ matrix, F, in which each row is normalized such that $\sum_j^N |F[i,j]| = 1 \quad \forall i \in M$. In a grid-based orientation histogram method such as SIFT, this would be a matrix with 16 rows and 8 columns, where each row is the normalized orientation histogram computed for a grid cell.

2. D is a $N \times N$ distance lookup matrix in which $D[i,j] = f(2^{-i}, 2^{-j})$, where f is a decomposed portion of a distance metric (see Section 2.2 for examples). As our compression method will use negative powers of two to approximate values in a normalized distribution, D need only contain distance computations for an appropriate range of those

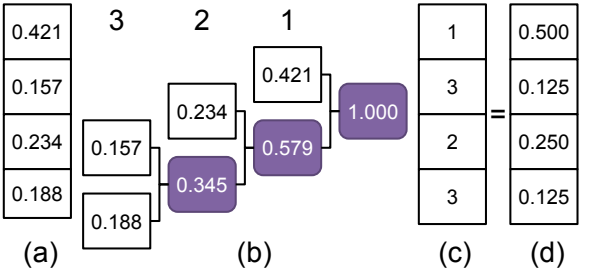

Figure 2: **Tree Coding** This figure displays the method by which a four-dimensional normalized histogram is compressed by way of tree coding. The original histogram, (a), is used to build a Huffman tree [8] (b). Each value in the original histogram is replaced by the depth of the Huffman tree at which it was placed, as seen in (c). These depths represent negative powers of 2, which are used to approximate the original histogram as seen in (d).

values.

3. A weight vector, **w**, of length M determining how much a particular row should contribute to the overall distance. For example, in the case of a center-weighted grid-based method, each index would correspond to the value of a central bi-variate Gaussian in grid coordinates computed for the corresponding grid cell.

First, the F matrix is computed for an image patch. Each row of this matrix is compressed using tree coding, as described in Section 2.1, resulting in a vector **c**. Each index of **c** encodes an entire row from F using $O(N \log N)$ bits, where N is the length of the original histogram. **c** vectors are used to index the distance matrix D to compute a distance between descriptors without requiring decompression, as described in Section 2.2. In Section 3 we will demonstrate how to convert the SIFT, SURF and GLOH descriptors into this canonical form.

2.1 Tree Coding

The normalized rows of F can be compressed using methods developed for the lossy compression of probability distributions. Gagie proposes two algorithms in [6] which can be used for this purpose with a bound on the Kullback-Leibler divergence [11] between the original and compressed distributions. Both of these algorithms use binary trees to assign a depth code to each element of the distribution which corresponds to a negative power of 2. The algorithm we use is based on Huffman trees [8], and as proven in [6] this encoding guarantees that $D(P||Q) < 1$, where P is the original distribution, Q is the compressed distribution, and $D(P||Q) = \sum_i p_i \log \frac{p_i}{q_i}$.

An example of the encoding process from [6] can be seen in Figure 2 for a four-dimensional vector. As the maximum depth of a tree is N, where N is the dimensionality of the original vector (*i.e.* the number of columns in F), $\log_2(N-1)$ bits are required to encode each index of the distribution.

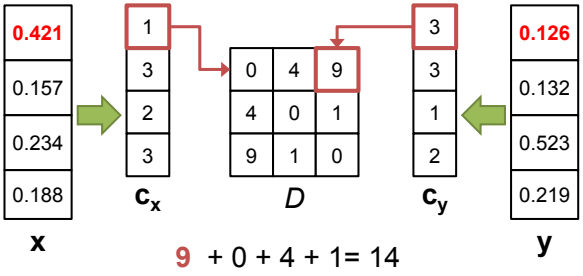

Figure 3: **Distance Computation** Given two four-dimensional normalized histograms $\mathbf{x}$ and $\mathbf{y}$, tree coding is used to create compressed forms $\mathbf{c_x}$ and $\mathbf{c_y}$. Each value of the compressed form indexes into the distance matrix D, computing the histogram distance as $\sum_j D[\mathbf{c_x}[i,j],\mathbf{c_y}[i,j]]$. In this case, D encodes the squared distance as part of a L_2 norm calculation (scaled by 64 to obtain integer values).

2.2 Distance Computation

One thing shared by most descriptor algorithms is some method of weighting the contributions of the sample points, usually by way of a bivariate normal distribution. The purpose of this weighting is to increase robustness to detector error by concentrating the distinctiveness of the descriptor at the center. Since it is common practice to treat the concatenated grid cell n-tuples as a vector and normalize it to have a length of one (to increase robustness to contrast and other imaging conditions), this weighting is maintained in the final descriptor. However, in our canonical form, each row in F is normalized separately to allow for compression. As a result the center weighting is lost and must be recaptured by way of the weighting vector $\mathbf{w}$.

Let $\mathbf{x}$ be a descriptor computed by a method in which M grid squares have individual normalized orientation histograms of length N whose contributions to the final descriptor are center-weighted, similar to that shown in Figure 1. Thus, $\mathbf{x}$ is a 32-dimensional vector in which each histogram is weighted by a specific weight depending on the location of its source cell in the grid, $\mathbf{x} = \{\mathbf{w}[1]h_{\mathbf{x},1,1}, \mathbf{w}[1]h_{\mathbf{x},1,2} \ldots \mathbf{w}[i]h_{\mathbf{x},i,j} \ldots \mathbf{w}[M]h_{\mathbf{x},M,N}\}$, where $\mathbf{w}[i]$ is the cell-specific weight and $h_{i,j}$ is the histogram value for bin j of histogram i. When comparing $\mathbf{x}$ with another vector $\mathbf{y}$ using the L_1 Minkowski distance, we get the following result:

$$L_1(\mathbf{x},\mathbf{y}) = \sum_i^M \sum_j^N |\mathbf{w}[i]h_{\mathbf{x},i,j} - \mathbf{w}[i]h_{\mathbf{y},i,j}| \quad (1)$$

$$= \sum_i^M \mathbf{w}[i] \sum_j^N |h_{\mathbf{x},i,j} - h_{\mathbf{y},i,j}| \quad (2)$$

As can be seen, the weight factor w can be separated from the distance computation. In this way, the distance lookup matrix D and the weighting vector $\mathbf{w}$ are linked, as the choice of metric encoded in D will determine the values in $\mathbf{w}$. Examples of this can be seen in Section 3.

Once each row of F has been compressed using tree coding, the result is the vector $\mathbf{c}$ in which each value encodes an entire row using $N\log_2(N-1)$ bits. Each value in $\mathbf{c}$ can be expanded and used to index into the distance matrix D. D is constructed in such a way that

it can replace the interior element of the distance computation:

$$L_1(\mathbf{c_x}, \mathbf{c_y}) = \sum_i^M \mathbf{w}[i] \sum_j^N \left| 2^{-\mathbf{c_x}[j]} - 2^{-\mathbf{c_y}[j]} \right| \tag{3}$$

$$L_1(\mathbf{c_x}, \mathbf{c_y}) = \sum_i^M \mathbf{w}[i] \sum_j^N D\left[\mathbf{c_x}[j], \mathbf{c_y}[j]\right] \tag{4}$$

Thus, $D[i,j] = f(2^{-i}, 2^{-j})$, where f is a decomposed element of a distance metric. In the case of Minkowski distances, $f(x,y) = (x-y)^m$. For the Jeffreys distance [14], $f(x,y) = x\log_2 \frac{x}{z} + y\log_2 \frac{y}{z}$, where $z = \frac{x+y}{2}$. The advantage of this method is that these values can be computed ahead of time, reducing the number of operations required for distance computations at runtime. Additionally, computing the distance between grid cell n-tuples separately before weighting results in an improvement in performance for all descriptor algorithms, as can be seen in Figure 7. An example of the method can be seen in Figure 3.

3 Case Studies

In order to demonstrate the strong performance and general applicability of our technique, we have performed case studies examining the conversion of three major local descriptor algorithms to the canonical form detailed in Section 2. Each of these algorithms consists of a collection of n-tuples each associated with a cell within a grid that is superimposed upon an image patch. As most successful descriptor algorithms follow this general pattern, the techniques we use to convert these algorithms to our canonical form should be of use for those who wish to apply our method to others.

3.1 SIFT

Perhaps the most widely-used local image descriptor is the descriptor portion of David Lowe's SIFT system [12], an acronym of **S**cale-**I**nvariant **F**eature **T**ransform. The SIFT descriptor consists of 16 orientation histograms, each with 8 bins, computed on a 4x4 grid. Each grid square contains 16 regularly space sample points, which are weighted by a centered Gaussian. The final descriptor is normalized to unit length, and descriptors are compared using L_2 norm. Our canonical form of SIFT, named NSIFT, is constructed as follows:

1. F is a 16x8 matrix where the orientation histograms are constructed in the same way as in Lowe's method, but each individually normalized to one after the incorporation of a Dirichlet prior.

2. D is a 8x8 matrix where $D[i,j] = x\log_2 \frac{x}{z} + y\log_2 \frac{y}{z}$ (the decomposed Jeffreys Divergence [14]) where $x = 2^{-i}$, $y = 2^{-j}$ and $z = \frac{x+y}{2}$.

3. $\mathbf{w}$ is a 16-dimensional vector where $\mathbf{w}[i] = \mathcal{N}(i \mod 4, \frac{i}{4} | \mu, \Sigma)$ where $\mathcal{N}$ is a bivariate normal distribution, $\mu = (1.5, 1.5)$ and Σ is a diagonal covariance with $\sigma = 1.5$.

The compressed version based upon this canonical form is named CSIFT, and the comparison in performance can be seen in Figure 4. It requires 48 bytes to encode (16 histograms, each requiring 24 bits) versus 512 bytes for the floating point representation we use in our experiments. In the pre-compressed NSIFT experiment, histograms are compared separately using the Jeffreys divergence as opposed to the L_2 norm.

3.2 SURF

The descriptor portion of the SURF algorithm of Bay *et al*. [2] (an acronym of **S**peeded-**U**p **R**obust **F**eatures) shares the 4x4 grid with SIFT, but eschews interpolation and orientation histograms, opting instead to use integral images to compute $\sum d_x$, $\sum d_y$, $\sum |d_x|$, and $\sum |d_y|$ for each cell, where d_x and d_y are Haar wavelet approximations. Each grid cell contains 25 regularly spaced sample points which are weighted using a centered Gaussian, resulting in a final 64 dimensional descriptor that is normalized to unit length. The resulting descriptors are compared using the L_2 norm.

We decided to work with the MU-SURF algorithm proposed by Agarwal *et al*. [1], due to its similar performance to USURF while being better suited for conversion. Our canonical form of SURF, named NSURF, consists of the following:

1. F is a 16x4 matrix, constructed in the manner of the MU-SURF descriptor, but without the second Gaussian weighting, with each row normalized to one.

2. D is an 8x8 matrix, where

$$D[i,j] = (g(i) - g(j))^2$$
$$g(x) = \begin{cases} 2^{-x} & x < 4 \\ -2^{4-x} & x \geq 4 \end{cases}$$

   This is to accommodate the fact that the first two indices of the cell vector can be negative.

3. **w** is the same as the NSIFT **w**, reflecting the second Gaussian weighting from the MU-SURF descriptor, except each element is squared due to the use of the L_2 norm in D.

Since the first two indices of the cell vector can be negative, they require an additional bit to encode in the compression step. This leads to 10 bits per histogram, for a total footprint of 20 bytes, versus 256 for the floating point representation used in our experiments. We name this form CSURF, and the comparison in performance can be seen in Figure 5.

3.3 GLOH

The **G**radient **L**ocation and **O**rientation **H**istogram (GLOH) descriptor of Mikolajczyk and Schmid [13] is an improvement upon the SIFT descriptor, replacing the regular 4x4 grid with a log-polar grid and increasing the number of dimensions in the orientation histogram to 16. This high-dimensional vector is then compressed using PCA to 128 dimensions. We propose that our method is a better fit in this case, as we can achieve a lower bitrate without losing the distinctiveness of the original form of the descriptor. In this case, the canonical form, NGLOH, is similar to NSIFT:

1. F is a 17x16 matrix where the orientation histograms are constructed in the same way as in Lowe's method, but each individually normalized to one after the incorporation of a Dirichlet prior.

2. D is a 16x16 matrix where $D[i,j] = x \log_2 \frac{x}{z} + y \log_2 \frac{y}{z}$ (the decomposed Jeffreys Divergence [14]) where $x = 2^{-i}$, $y = 2^{-j}$ and $z = \frac{x+y}{2}$.

3. Due to the radial nature of the grid, each ring of bins has the same weight. Thus, $\mathbf{w}$ is a 3-dimensional vector where $\mathbf{w}[i] = \mathcal{N}(i|\mu,\sigma)$, where $\mathcal{N}$ is a normal distribution, i is the ring the cell belongs to, $\mu = 0$ and $\sigma = 1$.

The compressed version of the canonical form is named CGLOH, and the comparison in performance can be seen in Figure 6. It requires 136 bytes to encode (17 histograms, each requiring 64 bits) versus 1088 for the floating point representation we use in our experiments. In the pre-compressed NGLOH experiment, histograms are compared separately using the Jeffreys divergence as opposed to the L_2 norm.

4 Experiments

Our experiments were performed on a random sample of 10,000 correct matches and 10,000 incorrect matches drawn from the Notre Dame patch correspondence dataset of Winder and Brown [17]. As in their experiments, we incorporated a "jitter" factor during testing. We altered each patch in the dataset using a random affine transform generated by drawing a scale offset, an angle offset, and x and y offsets from normal distributions with zero mean and standard deviations of .2, $\frac{\pi}{10}$ and .4 respectively (the scale offset is set to 2^{-x} where x is the value drawn from the distribution). These distributions approximate the statistics provided by Winder and Brown in their supplemental material, in which they measured the natural inaccuracies of a Harris-Laplace detector.

In each experiment, a particular algorithm is used to compute descriptors for each pair of patches. The metric score between the descriptors is computed in the manner appropriate to the algorithm, and used to create a Receiver Operating Characteristic curve which measures the effectiveness of that algorithm. The curve is computed using a moving threshold, as in [17]. In addition to the reference implementations of SIFT, SURF and GLOH we also include results from experiments using a zero-normalized 30x30 patch compared using cross correlation (ZNCC) as a baseline. ROC curves displaying the performance of our canonical form and compressed descriptors for SIFT, SURF and GLOH can be seen in Figures 4, 5 and 6 respectively. Figure 7 displays the error rates at 95% detection for all experiments.

5 Discussion

As can be seen in Figure 7, our compression method not only reduces the size of standard descriptors significantly, but by normalizing each cell vector separately it can exploit metrics such as the Jeffreys divergence to achieve a significant improvement over the original in appropriate cases, such as for SIFT and GLOH. Due to tree coding, lookup tables with these more computationally expensive metrics can be pre-computed for the appropriate powers of two such that matching of compressed descriptors using these metrics is less computationally expensive than matching the original descriptor using the L_2 norm. However, as the SURF case study shows, descriptors which are not amenable to this technique see less dramatic improvement for the canonical and compressed forms over the original. Regardless, in each case our canonical form is an improvement over the original descriptor, and compression from this form results in little or no loss of performance.

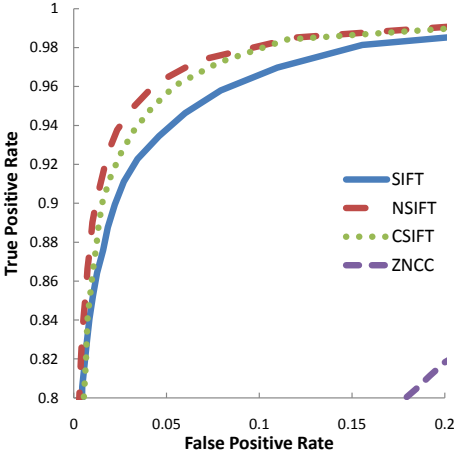

Figure 4: **SIFT Performance** ROC curves for SIFT (512 bytes), NSIFT (canonical form) and CSIFT (48 bytes).

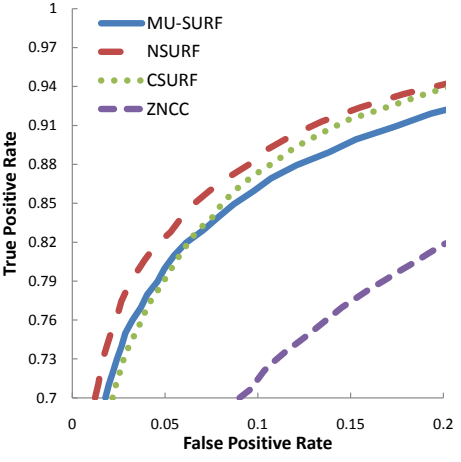

Figure 5: **SURF Performance** ROC curves for SURF (256 bytes), NSURF (canonical form) and CSURF (20 bytes).

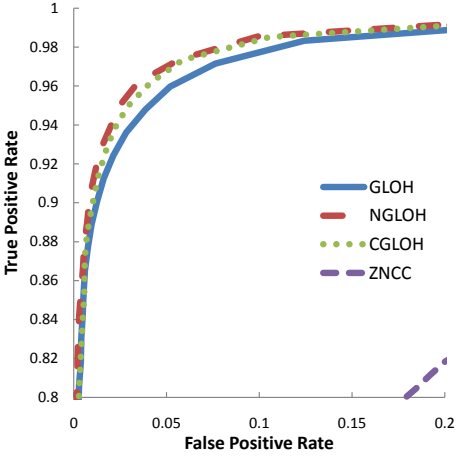

Figure 6: **GLOH Performance** ROC curves for GLOH (1088 bytes), NGLOH (canonical form) and CGLOH (136 bytes)

ZNCC	Reference	52.2(3600)
SIFT	Reference	7.29(512)
NSIFT	Canonical	3.27(512)
CSIFT	Compressed	4.29(48)
MU-SURF	Reference	30.5(256)
NSURF	Canonical	23.0(256)
CSURF	Compressed	22.8(20)
GLOH	Reference	4.09(1088)
NGLOH	Canonical	2.47(1088)
CGLOH	Compressed	2.91(136)

Figure 7: **Error Rates at 95% Detection** The number in parenthesis is the bytes used to encode the descriptor in our experiments.

6 Conclusion

The generalized descriptor compression method proposed in this paper not only significantly reduces the size of feature descriptors for easier storage, but reduces the cost of matching descriptors during database search and retrieval. We have described how to convert SIFT, SURF and GLOH to our canonical descriptor form, and experimentally shown that their performance in this canonical form is better. Additionally, we have shown that the compressed versions of these descriptors improve on their original performance while requiring far fewer bits to encode and fewer processor operations during the matching stage.

References

[1] Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. CenSurE: Center surround extremas for realtime feature detection and matching. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Proc. European Conf. on Computer Vision*, volume 5305 of *LNCS*, pages 102–115. Springer, 2008.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[3] V. Chandrasekhar, G. Takacs, D. Chen, S. Tsai, and B. Girod. Transform coding of feature descriptors. In *Proc. of SPIE Conf. on Visual Communications and Image Processing*, 2009.

[4] V. Chandrasekhar, G. Takacs, D. Chen, S. Tsai, R. Grzeszczuk, and B. Girod. CHoG: Compressed histogram of gradients. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2009.

[5] N. Davies, A. Hesse, A. Dix, and K. Cheverst. Understanding the role of image recognition in mobile tour guides. In *Proc. of the Mobile HCI*, Salzburg, Austria, 2005.

[6] T Gagie. Compressing probability distributions. *Information Processing Letters*, 97 (4):133–137, February 2006.

[7] S. W. G. Hua, M. Brown, and S. Winder. Discriminant embedding for local image descriptors. In *Proc. Int. Conf. on Computer Vision*, pages 1–8. IEEE Computer Society, 2007.

[8] D.A. Huffman. A method for the construction of minimum-redundancy codes. In *Proc. of the I.R.E.*, pages 1098–1102, 1952.

[9] Menglei Jia, Xin Fan, Xing Xie, Mingjing Li, and Wei-Ying Ma. Photo-to-search: Using camera phones to inquire of the surrounding world. *Mobile Data Management, IEEE International Conference on*, 0:46, 2006. ISSN 1551-6245.

[10] Y. Ke and R. Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 2, pages 506–513. IEEE Computer Society, 2004.

[11] S. Kullback. *Information Theory and Statistics*. Dover, New York, NY, 1968.

[12] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Computer Vision*, 60(2):91–110, November 2004.

[13] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10(27):1615–1630, 2005.

[14] Jan Puzicha, Thomas Hofmann, and Joachim M. Buhmann. Non-parametric similarity measures for unsupervised texture segmentation and image retrieval. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 267–272, 1997.

[15] G. Shakhnarovich and T. Darrell. *Learning Task-Specific Similarity*. PhD thesis, Massachusetts Institute of Technology, 2005.

[16] 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. In *Proc. ACM Int. Conf. on Multimedia Information Retrieval*, pages 427–434, 2008.

[17] S. A Winder and M. Brown. Learning local image descriptors. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[18] T. Yeh, K. Tollmar, and T. Darrell. Searching the web with mobile images for location recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 2, pages 76–81, 2004.

[19] C. Yeo, P. Ahammad, and K. Ramchandran. Rate-efficient visual correspondences using random projections. In *Proc. of IEEE Int. Conf. on Image Processing*, 2008.