Transform Coding of Image Feature Descriptors

Vijay Chandrasekhar[1], Gabriel Takacs[1], David Chen[1],
Sam S. Tsai[1], Jatinder Singh[2], Bernd Girod[1] *

[1]Stanford University, Stanford, CA, USA
[2]Deutsche Telekom Laboratories, Los Altos, California, USA

ABSTRACT

We investigate transform coding to efficiently store and transmit SIFT and SURF image descriptors. We show that image and feature matching algorithms are robust to significantly compressed features. We achieve near-perfect image matching and retrieval for both SIFT and SURF using ∼2 bits/dimension. When applied to SIFT and SURF, this provides a 16× compression relative to conventional floating point representation. We establish a strong correlation between MSE and matching error for feature points and images. Feature compression enables many application that may not otherwise be possible, especially on mobile devices.

Keywords: image matching, robust features, feature descriptor compression, transform coding

1. INTRODUCTION

Local image features have become pervasive in the areas of computer vision (CV) and image retrieval (IR). Robust local descriptors, such as Scale-Invariant Feature Transform (SIFT),[1] Gradient Location and Orientation Histogram (GLOH),[2] and Speeded-up Robust Features (SURF),[3] are used to overcome image variability caused by changing viewpoints, occlusions, and varying illumination.

There are two main benefits of feature compression. (1) *Compact server-side storage*: Image retrieval requires a query image to be matched against databases of millions of features on the server. Feature compression can yield tremendous savings in disk space usage. (2) *Reduced network transmission*: Compressed features can be more efficiently transmitted over a network. For example, if a mobile camera transmits the features extracted from a query image to a remote server over a wireless connection, feature compression is crucial for keeping the transmission time low.

Research on robust local descriptors continues to be a very active area of computer vision. Of the many proposed descriptors, SIFT[1] is probably the most commonly used. Other popular descriptors include GLOH by Mikolajczyk and Schmid,[4] and SURF by Bay *et al.*[3] The review paper by Mikolajczyk *et al.*[2] provides a comprehensive analysis of several descriptors. Winder and Brown[5] 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. Traditionally, feature data are reduced by decreasing the dimensionality of descriptors. Ke and Sukthankar[6] investigate dimensionality reduction via Principal Component Analysis. Hua *et al.*[7] propose a scheme that uses Linear Discriminant Analysis. Takacs *et al.*[8] quantize and entropy code SURF feature descriptors for reducing their bit rate. Chuohao *et al.*[9] 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. In their recent work, Torralba *et al.*[10] use a machine learning technique called Similarity Sensitive Coding[11] to train binary codes on images. Shakhnarovich, in his thesis,[12] uses a similar approach to train binary codes on image patches.

In this work, we investigate a lossy transform-based feature compression scheme that is rate-distortion optimized. We provide random access of compressed features from a database by independently compressing each feature. We show that image and feature matching algorithms are robust to significantly compressed features. Our algorithm has been tested on a standard image dataset, using both SIFT and SURF descriptors. With these data, our algorithm provides a 16× compression relative to a single-precision floating-point representation, while

* Please send correspondence to Vijay Chandrasekhar (vijayc@stanford.edu).

introducing negligible image matching error. Additionally, we show that MSE is a good predictor for both image and feature match error.

In Sec. 2, we present an overview of how SIFT and SURF descriptors are computed. We also discuss the statistics of individual dimensions of these descriptors. In Sec. 3, we present our feature compression algorithms. Then, in Section 4 we provide a mathematical analysis of the algorithm. In Sec. 5, we provide experimental results showing the performance of our scheme.

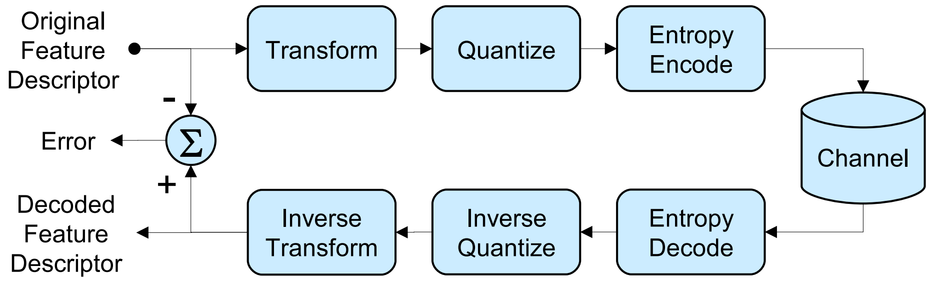

Figure 1. System block diagram, showing the encoder (*top*), channel (*right*), and decoder (*bottom*). The error in the reconstructed descriptor (*left*) reflects the distortion induced by feature compression.

2. DESCRIPTOR OVERVIEW

Lowe,[1] first showed how computing the distribution of gradients was an effective strategy in defining descriptors on patches. SIFT and SURF are Histogram of Gradients (HoG) descriptors that capture the distribution of gradients within a collection of pixels. HoG-based descriptors tend to be high dimensional, and are robust to geometric deformations and small localization errors in scale space.

To compute descriptors, image patches detected around keypoints are first divided into different cells — SIFT and SURF use 4×4 square grid configurations. Following this, HoG descriptors compute the d_x and d_y gradients within each cell. For SIFT, the gradient orientation within each cell is quantized to 8 angles, and the gradient magnitude along each direction is summed. While SIFT quantizes gradients by binning along angular directions, the SURF descriptor consists of Σd_x, Σd_y, $\Sigma|d_x|$, $\Sigma|d_y|$ for each cell.

SIFT and SURF descriptors can be represented as functions of the gradient Probability Mass Function (PMF). Let $P_{D_x,D_y}(d_x,d_y)$ be the joint (x,y)-gradient distribution in a cell, and N be the number of pixels in the cell. Note that the gradients within a cell are typically weighted by a Gaussian prior to descriptor computation.[1,3]

The 8 SIFT components of a cell, $\mathcal{D}_{SIFT}$, can be computed as

$$\mathcal{D}_{SIFT}(i) = N \sum_{\omega \in \Omega} A(d_x,d_y) P_{D_x,D_y}(d_x,d_y) \qquad (1)$$

where $\Omega = \{\frac{\pi i}{4}\}, i = 1 \dots 8$, and $\omega = \tan^{-1}\frac{d_y}{d_x}$, and $A = \sqrt{d_x^2 + d_y^2}$.

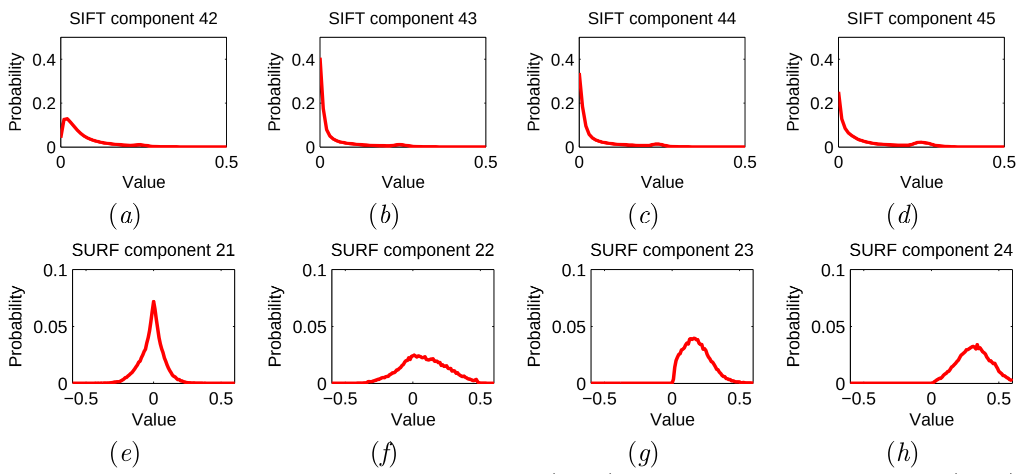

Figure 2. Distributions of SIFT components 42-45 (a - d) and SURF components 21-24 (e - h).

The 4 SURF components of a cell, $\mathcal{D}_{SURF}$, can be computed as

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

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

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

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

For SURF, the relationship between components $D_{SURF}(1)$ and $D_{SURF}(3)$, and similarly, components $D_{SURF}(2)$ and $D_{SURF}(4)$ in each cell is exploited in one of the coding schemes proposed in Section 5.2.

In Figure 2, we show the distributions of components 42-44 and components 21-24 of SIFT and SURF descriptors respectively. These components of the descriptor correspond to the gradient information in one of the inner cells of the 4×4 grid. Note that all SIFT components are positive, while half the SURF dimensions are restricted to positive values.

Note that the variances of the different dimensions of the descriptor are different. The gradients are weighted by a Gaussian with a higher weighting assigned to pixels that are closer to the detected keypoint. As a result, the components of the descriptor corresponding to the inner cells have higher variances than the components corresponding to the outer cells. Additionally, image patches are rotated in the direction of the most dominant gradient before descriptor computation. As a result, the variances of components corresponding to $\sum d_y$ and $\sum |d_y|$ are higher than those corresponding to the components $\sum d_x$ and $\sum |d_x|$. The different variances of each component are taken into account when computing the coding gain in Section 4.1.

3. COMPRESSION ALGORITHM

In this section, we describe the design of the feature descriptor codec. Since image features are unordered and distinct from each other, we have designed our codec to operate on each descriptor individually. Independent encoding gives the advantage of having random access to a given feature without the need to reference other features. The goal of our codec is to compress each feature individually by using intrinsic descriptor statistics. Fig. 1 shows the block diagram for the proposed codec, consisting of three main parts: the encoder, the channel, and the decoder. The channel could be a wireless network separating a mobile camera and a server.

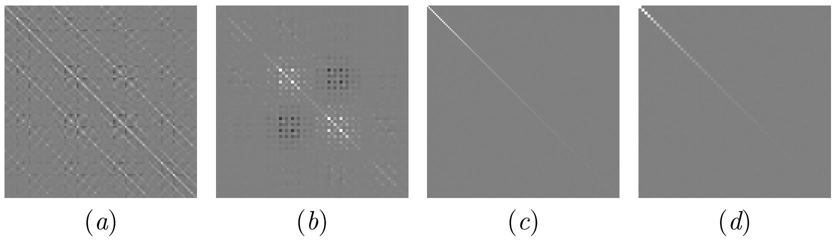
Figure 3. Covariance matrices for 128-dimensional SIFT (a) and 64-dimensional SURF (b) descriptors, and covariance matrices for transformed SIFT (c) and SURF (d) descriptors. Mid-gray represents zero.

3.1 Encoder

The encoder first applies a linear transform to the feature descriptor. The transform must be invertible so that the feature can be recovered at the decoder. Additionally, as described in Section 4, we desire an orthonormal transform that reduces correlation between descriptor elements and compacts energy into a smaller number of dimensions. We choose the Karhunen-Loeve Transform (KLT) because it is the orthonormal transform that optimally decorrelates different dimensions of a feature vector and compacts energy into the first few dimensions. The KLT, however, is data-dependent and must be tailored to the statistics of the features on which it is applied. However, the calculation of the KLT can be performed once for a large training set, and then the KLT is stored for use at the encoder and decoder.

After the transform, we apply scalar quantization in each dimension of the transformed feature vector. By varying the step-size of the quantizer, the codec can trade off bit rate against distortion in the reconstructed feature vector. Finely quantized features require many bits to encode, while coarsely quantized features require fewer bits. The final stage before transmission or storage is entropy coding. We use an arithmetic coder because it can achieve rates close to entropy.

3.2 Decoder

Upon receipt of the compressed feature data, the decoder performs the inverse operations. First, the entropy-coded symbols are decoded into quantization levels. These levels are then mapped to real-valued numbers to form the transform coefficients. The inverse transform is applied to the coefficients to recover the decompressed feature. For the purposes of analysis, the decoded feature can then be compared to the original feature to compute the distortion.

4. RATE-DISTORTION ANALYSIS

In this section, the compression performance of the feature codec proposed in Section 3 is analyzed. Use of the KLT prior to quantization is justified based on the transform's optimal energy compaction and decorrelation properties.

4.1 Coding Gain

The SIFT and SURF descriptors do not attempt to minimize correlation between the individual descriptor dimensions. Figure 3 (a, b) shows the covariance matrix of the SIFT and SURF descriptors. The large magnitudes of particular off-diagonal terms indicate that some dimensions are strongly correlated with other dimensions. The correlation is due to the way that the descriptor components are computed from gradients and their absolute values, each weighted by a two-dimensional Gaussian.

Our codec applies the KLT because the KLT provides two important benefits. First, the KLT decorrelates feature dimensions, as seen in Figure 3 (c, d). The strong correlation between different dimensions from before has now been removed. Second, of all orthonormal transforms, the KLT maximally compacts energy in the largest-energy dimensions. The transform coding gain G_T can then be computed as follows:

$$G_T = \frac{\left(\prod_{i=1}^{N} \sigma^2_{pre,i}\right)^{\frac{1}{N}}}{\left(\prod_{i=1}^{N} \sigma^2_{post,i}\right)^{\frac{1}{N}}} \quad , \tag{6}$$

where $\sigma^2_{pre,i}$ is the variance of the i^{th} dimension of the original vectors, $\sigma^2_{post,i}$ is the variance of the i^{th} dimension of the KLT vectors, and N is the dimensionality of the descriptor (64 for SURF, 128 for SIFT). The numerator and denominator in Equation (6) can be interpreted as the ratio of the geometric means, respectively, of the dimensional variances of the pre and post transform descriptors. Note that the ratio of geometric means is considered because the variances of the different dimensions pre-transform are not equal due to the descriptor properties discussed in Section 2. The theoretical transform coding gains for SIFT and SURF are calcuated to be 2.6 and 1.7 respectively. Section 5.2 discusses the actual coding results.

Compression performance can be measured in terms of the minimum distortion, D, achievable at a bit rate, R. For example, D can be MSE or matching error. Suppose a reference codec applies no transform and optimally chooses scalar quantization step sizes $\{\Delta_i^{RE}\}_{i=1}^N$, to achieve the minimum distortion, D^{RE}, at bit rate R. Similarly, assume a transform-based codec chooses quantization step sizes $\{\Delta_i^{TF}\}_{i=1}^N$, corresponding to the dimensions of the transformed feature, to minimize D^{TF} at bit rate R. Then, the ratio of the two distortions is the measured coding gain,

$$\widehat{G_T} = \frac{D^{RE}}{D^{TF}} \quad , \tag{7}$$

from which we see coding gain is the factor by which distortion can be decreased relative to the reference codec by utilizing a transform. Following the KLT, we apply uniform scalar quantization in each dimension.[13]

Although minimizing MSE is a desirable goal, the ultimate goal is to minimize the matching error. If we assume that the matching error increases monotonically with MSE, then minimization of the MSE also implies minimization of the matching error. Experimentally, we have verified that this assumption is valid over a wide range of bit rates. Thus, our method of rate-distortion tradeoff also works well regardless of whether MSE or matching error shall be minimized.

5. EXPERIMENTAL RESULTS

To evaluate our scheme we have performed experiments with a well-known dataset. We have evaluated the distortion induced by compression as well as its effects on feature and image matching. In Section 5.1, we describe the experimental data sets used. In Section 5.2, we present the rate distortion performance of transform coding SIFT and SURF descriptors. In Section 5.3, we evaluate the effect the compression on feature match rate, and image rate by considering ground truth image pairs. Finally, in Section 5.4, we evaluate the effect of transform coding on image retrieval from a database using a Scalable Vocabulary Tree (SVT).[14]

5.1 Experimental Data Set

We have chosen the *ZuBuD* dataset[15] for evaluation of our scheme. This dataset consists of an image database and a set of query images, both taken of buildings in Zürich. The database contain 5 views each of 201 buildings. There are 115 query images which are not contained in the database. We extracted both SIFT-128 and SURF-64 features from all of the images, yielding about 1500 and 500 features per image, respectively. For training, we adapted our compression algorithm to the statistics of the database features, and then evaluated the performance by compressing the query features.

5.2 Rate Distortion Performance

To determine the trade-off between bit rate and distortion, we have compressed and decompressed all the feature descriptors from the query images while varying the quantization step-size. Each step-size yields a rate/distortion pair. We use the peak signal-to-noise ratio (PSNR) as a metric for distortion, which for feature descriptors normalized to a maximum magnitude of 1 is given in dB as

$$\text{PSNR} = -10 \cdot \log_{10}(\text{MSE}) \tag{8}$$

Fig. 4 shows the rate-distortion performance of our codec on SIFT and SURF descriptors on the left and right, respectively.

For SURF, we compare the transform coding scheme *KLT* to two other schemes. The first scheme, *No KLT*, applies no transfrom and uses uniform scalar quantization of each dimension of the descriptor. The second scheme, *Joint*, also applies no transform, and additionally, exploits the relationship between the different dimensions in each individual cell. The *Joint* scheme creates symbol pairs for dimensions $[D_{SURF}(1), D_{SURF}(3)]$, and $[D_{SURF}(2), D_{SURF}(4)]$ from Equation 2, and encodes them with an arithmetic coder.

The *KLT* scheme performs better than *Joint* and *No KLT* at all bit rates. We observe that the coding gain of 1.76 is realized at high rates. The difference between the *KLT* and *No KLT* curves at high rates is correctly predicted to be 0.4 bits/dimension by Equation 7. We show in Section 5.3 that the operating points at 1.5 and 2.5 bits/dimension provide a good margin on image and feature match errors. At the operating points of 1.5 and 2.5 bits/dimension, the $\hat{\text{K}}$LT provides 20% and 10% gains, respectively, in bit rate compared to *No KLT*. At high rates, the performance of the *Joint* scheme is close to the *KLT* scheme.

Similarly for SIFT, we compare the transform coding scheme to a scheme *No KLT* that applies no transform and uses uniform scalar quantization in each dimension. We observe that transform coding performs better at low bit rates. At the recommended operating points of 1.5 and 2.5 bits/dimension, the two schemes *KLT* and *No KLT* perform on par. At high rates, using the KLT hurts the compression performance, and the predicted coding gain of 2.6 is not realized. The KLT gives a good rotation for scalar quantization for Gaussian statistics as the decorrelation aligns the Gaussian distribution with the symbol axes. However, the statistics for SIFT features are not Gaussian as shown in Figures 2. Also, the independence of transform coefficients does not result from decorrelation if the source does not follow Gaussian statistics. Feng and Effros[16] also show that optimal energy compaction with a KLT is neither unique nor sufficient for optimal transform coding. Applying the 128×128 transform to SIFT features converts the non Gaussian distributions shown in Figure 2 into Gaussian distributions that are more difficult to encode.

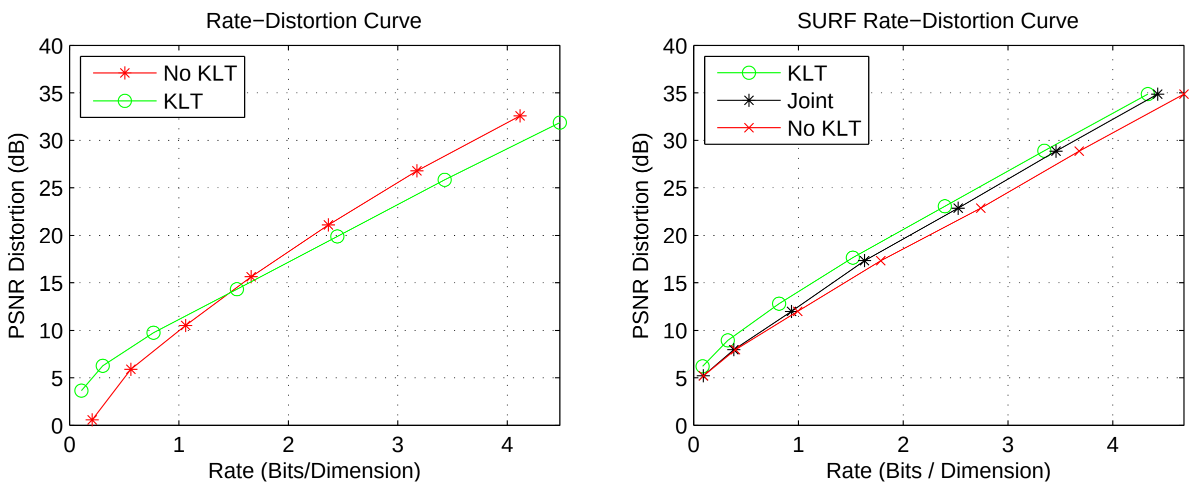

Figure 4. Distortion versus bit-rate for SIFT (*left*) and SURF (*right*) descriptors. Distortion is measured in peak signal-to-noise ratio (PSNR). Better performance is indicated by low bit-rate and high PSNR.

5.3 Matching Performance

Since the primary use of feature descriptors is image matching, we evaluate the effects of compression on matching rate. Each query image in the *ZuBuD* dataset has 5 matching images in the database, yielding 575 matching image pairs. For image matching, we apply the methods proposed by Brown and Lowe.[17] We match features using a ratio-test with a threshold of 0.8. Potential feature matches are then verified by geometric consistency checks (GCC) with RANSAC (RANdom SAmple Concensus). We compare the compressed feature match rate to ground truth computed with non-compressed features. As an intermediate result between MSE and image match rate, we also determine the feature match rate, given by

$$\rho_i = \frac{1}{M}\sum_{j=1}^{M}\frac{\left|F_{Q_i}^j \cap F_o^j\right|}{\left|F_o^j\right|} \tag{9}$$

where ρ_i is the feature match rate for the i^{th} quantization step-size and M is the number of image pairs. The set of unquantized features for the j^{th} image that passes GCC is given by F_o^j, and $F_{Q_i}^j$ is the set of matching features for the i^{th} quantization step-size. The vertical bars represent the cardinality of the set.

Fig. 6 shows the results of image matching with compressed features for SIFT and SURF on the left and right, respectively. At each rate, we pick the best scheme for compression based on the results in Section 5.2. In addition to image and feature match rates, Fig. 6 also shows the MSE distortion for comparison. Note that the feature matching rate is more sensitive to MSE distortion than is the image matching rate. The image matching rate does not degrade significantly until there are fewer than 50% feature matches.

For both SIFT and SURF, the error rate is small for feature and image matches at 1.5-2.5 bits per dimension,. While even lower bit-rates are possible, this operating point provides a comfortable margin of robustness. This gives almost 16$\times$ compression to conventional floating point representation, while providing low image match error. This indicates that image matching is quite robust to feature descriptor quantization. It is also important to note that both image and feature match rates monotonically decrease with MSE over a wide range of bit-rates. This means that optimizing our codec for MSE reduction will have a similar effect on match rate.

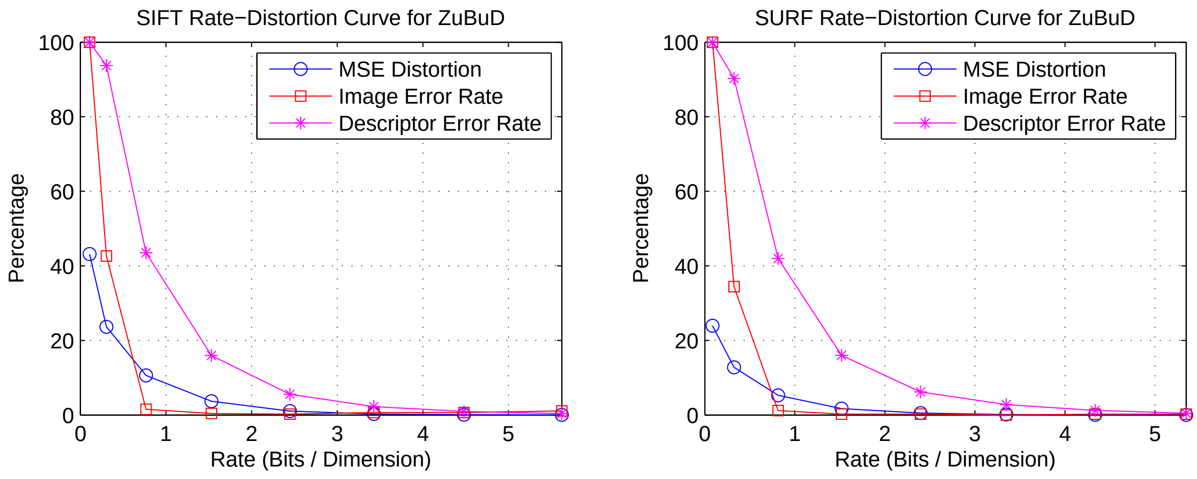

Figure 5. Error percentage versus bit-rate for SIFT (*left*) and SURF (*right*) descriptors. Smaller error percentages are better.

5.4 Retrieval Performance

To avoid an expensive linear search through the database, local features are typically hierarchically organized into a Scalable Vocabulary Tree (SVT).[14] The SVT can be used to narrow down the search to a small number of sufficiently similar database images. The best image is then decided by a comparison of the query image features against the features extracted from each of the sufficiently similar images.

The nodes of an SVT are the centroids determined by hierarchical k-means clustering of database feature descriptors. There are two parameters used in defining the hierarchical quantization, the branch factor, B, and tree depth, D. Initially, k-means is run on all of the training data with B cluster centers, after which the data is clustered into B groups. The quantization cells are further divided by recursively applying the process until D levels of the tree are obtained.

Suppose an SVT has a total of N interior and exterior nodes. During the training phase, all feature descriptors extracted from the i^{th} database image are classified through the SVT using a greedy nearest-neighbor search. After classification, it is known how many times, $n_i(j)$, the j^{th} node in the SVT is visited by the i^{th} database feature set resulting in a vector of node visit counts $d_i = [n_i(1), n_i(2), ...n_i(N)]$. Similarly, when a query feature set is received on the server, its descriptors are classified through the SVT generating a vector $q =$

$[n_q(1), n_q(2), ...n_q(N)]$. The dissimilarity between the i^{th} database feature set and the query feature set is given by

$$D_i\left(d_i, q\right) = \left\| \frac{Wd_i}{\|Wd_i\|} - \frac{Wq}{\|Wq\|} \right\|, \tag{10}$$

where $W = diag(w_1, ...w_N)$ is a matrix used to assign different entropy-related weights to different nodes in the tree. The norm could be L_1 or L_2. The minimum-D_i database feature sets are returned as the closest matching images.

For this experiment, we use the 1005 images from the *ZuBuD* dataset to build the SVT. The retrieval results are averaged over 115 query images in the *ZuBuD* dataset. We use SURF features for this experiment, however, similar results can be obtained for SIFT features. Each image typically has 500-1000 features resulting in about one million features for all *ZuBuD* images. The SVT is built with uncompressed database features. We pick a branch factor of $B = 10$ and depth $D = 6$, leading to one million leaf nodes.

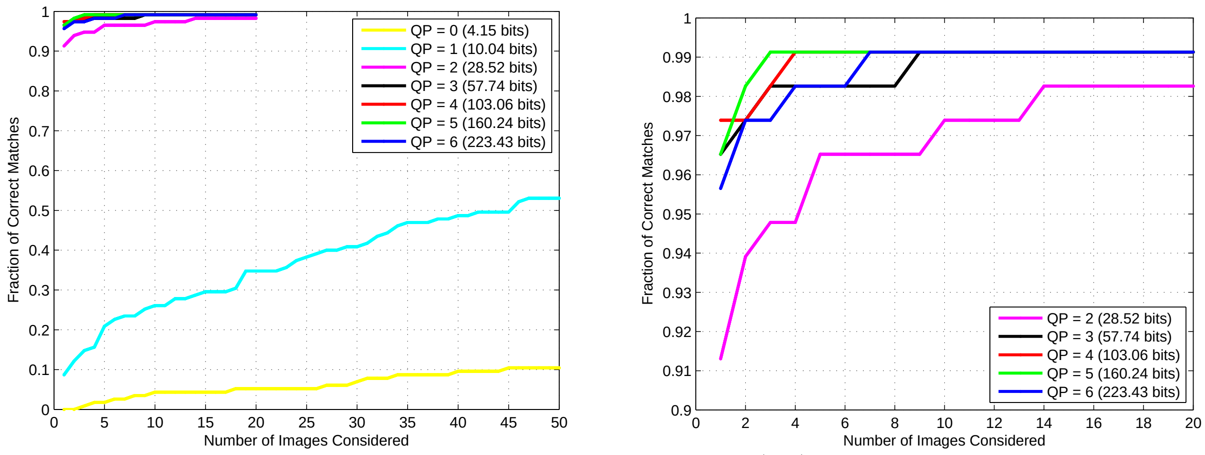

Figure 6. Effect of SURF feature compression on retrieval performance (*left*), and a detail of the high-rate curves (*right*). We plot the fraction of correct matching images among the top 20 images retrieved from the database. Good retrieval performance is achieved even at bit rates as low as 57 bits/feature.

Query features from each query image are compressed at different rates with transform coding. In Fig. 6, we plot the fraction of correct images among the top N images retrieved from the database as a function of N. The performance improves as the number of bits/feature increases from 4 to 223. Notably, for 99% of the query images, we can retrieve the correct matching image among the top 10 retrieved database images with only 57 bits/feature.

6. CONCLUSION

We investigated transform coding to efficiently store and transmit SIFT and SURF image descriptors. For SURF, a moderate transform gain is realized using a Karhunen-Loeve Transform. For SIFT, the KLT only achieves a gain at low bit-rates, since the non-Gaussianity of individual SIFT components is replaced by near-Gaussian transform coefficients. We show that image and feature matching algorithms are robust to coarsely quantized and compressed features. We achieve near-perfect image matching and retrieval for both SIFT and SURF below 2 bits/dimension. When applied to SIFT and SURF, this provides a 16× compression relative to conventional floating point representation. While even lower bit-rates are possible, this operating point provides a comfortable margin of robustness. We have established a strong correlation between MSE and match rates at both the image and feature levels. Practical feature compression enables many application that may not otherwise be possible, especially on mobile devices.

REFERENCES

1. D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision* **60**(2), pp. 91–110, 2004.
2. 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* **65**(1-2), pp. 43–72, 2005.
3. H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: Speeded Up Robust Features," in *ECCV (1)*, pp. 404–417, 2006.
4. K. Mikolajczyk and C. Schmid, "Performance Evaluation of Local Descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.* **27**(10), pp. 1615–1630, 2005.
5. S. A. Winder and M. Brown, "Learning local image descriptors," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on, Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on* , pp. 1–8, 2007.
6. 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)*, **02**, pp. 506–513, IEEE Computer Society, 2004.
7. 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.
8. 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.
9. P. A. Chuohao Yeo and K. Ramchandran, "Rate-efficient visual correspondences using random projections," *IEEE International Conference on Image Processing* , 2008.
10. A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large image databases for recognition," in *CVPR*, 2008.
11. 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*, p. 750, IEEE Computer Society, (Washington, DC, USA), 2003.
12. G. Shakhnarovich and T. Darrell, "Learning task-specific similarity," *Thesis* , 2005.
13. D. S. Taubman and M. W. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards and Practice*, Springer, 2002.
14. D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Conference on Computer Vision and Pattern Recognition*, pp. 2161–2168, (New York, NY, USA), June 2006.
15. H. Shao, T. Svoboda, and L. V. Gool, "Zubud-zurich building database for image-based recognition," in *Technical Report 260*, (Zurich, Switzerland), 2003.
16. H. Feng and M. Effros, "On the rate-distortion performance and computational efficiency of the karhunen-loeve transform for lossy data compression," *Image Processing, IEEE Transactions on* **11**, pp. 113–122, Feb 2002.
17. M. Brown and D. Lowe, "Unsupervised 3d object recognition and reconstruction in unordered datasets," in *International Conference on 3D Imaging and Modelling*, pp. 56–63, (Ottawa, Canada), June 2005.