

# Compression of Image-Based Features

Ivan Janatra, June Zhang

Final Project, EE398 (Image and Video Coding), Stanford University

**Abstract** – Image-based features are now commonly used to perform matching between objects in a query image and objects in database images. Previous studies have been done to apply these features in a real-time cell-phone-based mobile augmented reality (MAR) to recognize buildings. The database feature sets need to be transmitted from the database to the cell phone. In order to minimize cost associated to this transmission, a compression scheme should be developed. This document investigates the effectiveness of several possible compression algorithms. Initially, principal component analysis (PCA) is applied to maximize the coding gain. The transformed coefficients are quantized using a uniform quantizer and entropy-coded using several known entropy coding schemes. By varying the quantization level, the relationship between achievable bit rate, distortion, and query-to-database image matching rate could be determined.

**Index Terms** – Image-based Features, Data Compression, Mobile Augmented Reality (MAR)

## I. INTRODUCTION

Mobile device technology has now become a platform that supports numerous forms of multimedia, the most prevalent of which is image data. Moreover, some of these devices are capable of capturing scenes and storing them in digital formats like the JPEG file, at a resolution sufficiently fine for image recognition. Image recognition can then be done by feature matching. By characterizing the features available in a query image, one could do a search on a feature sets database to find a set of database features that match the query features. The mobile device can then augment the user’s reality by supplying information about the subject. For example, if the image is a historical building in a city, the information would be the name of the building and its history.

In this project, we investigate the effectiveness of several possible compression schemes to be applied to feature sets. Our data set comprises of 133 database images of buildings at Stanford University, and the corresponding query images, which are images of the same buildings taken at different time and orientation. We apply principal component analysis (PCA) to maximize the energy concentration, followed by uniform quantization. Finally we investigate the effectiveness of several known entropy coding schemes such as Huffman coding and arithmetic coding in encoding the image-based features. Our result is an established relationship between achievable bit rate, mean squared error distortion, and query-to-database image matching rate.

## II. IMAGE-BASED FEATURES

### A. Generation of Feature Sets

Given an image, the features are found using the Speeded-Up Robust Features (SURF) algorithm described by Bay, Tuytelaars, and Gool in [1]. There are two steps in generating the features. First, ‘interest points’ are selected at distinctive locations in the image, such as corners, blobs, and T-junctions. It is important for these interest points to be repeatable, i.e. the same interest points should be found under different viewing conditions. Next, the neighborhood of every interest point is represented by a feature vector or ‘descriptor’. The descriptor has to be distinctive and robust to noise and detection error.

Generation of feature descriptor consists of several sub-steps. The first step is to find a dominant orientation of the interest point. This is calculated using Haar-wavelet responses in x and y direction within a circular region around the interest point. Once this is done, a square region is constructed around the interest point taking into account the dominant orientation previously found. This square region is then divided to 4x4 square sub-regions. Each sub-region will have four descriptors: sum of Haar wavelet responses in x’ and y’ direction ( $\sum d_x, \sum d_y$ ), and sum of absolute value of the responses ( $\sum |d_x|, \sum |d_y|$ ). This brings a total of 64-dimensional feature vector for one interest point.

### B. Feature Matching Algorithm

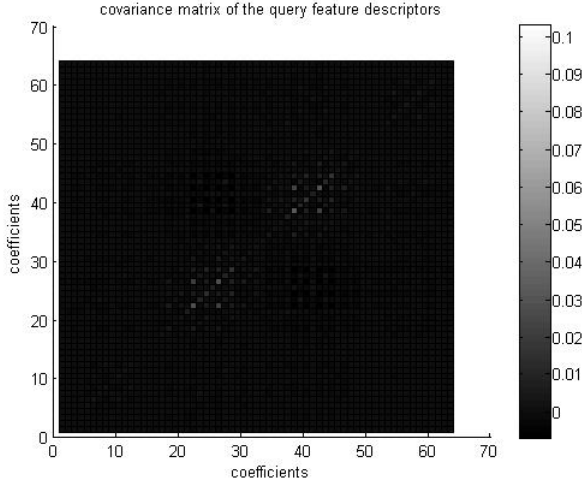
Image-based feature matching scheme used in this project is detailed in [2]. The main components are ratio test and geometry consistency check. Feature correspondences for pair of images are calculated using the nearest-neighbor ratio matching strategy. When comparing test image  $I_t$  to image  $I_m$ , for each feature descriptor  $F_t^i$  in  $I_t$  we find two features in  $I_m$  whose descriptor values are closest to  $F_t^i$  in a sum-of-squared-difference sense. If the ratio of their distances is less than 0.8, we decide that  $F_t^i$  matches its most similar feature from  $I_m$ , otherwise we decide that  $F_t^i$  doesn’t have a match. Using ratio test has been shown to be more robust against incorrect matches.

Geometry consistency check is performed to remove incorrect pairings produced by the ratio test. The idea is that if feature matching is perfect, there should be an affine transformation that transforms the location of one feature in query image to the location of the corresponding feature in the database image. Modified RANSAC algorithm is used to find an affine model consistent with the majority of feature matches. Moreover, additional constraints are imposed to

eliminate implausible transformations, such as reflection and excessive shear, and data points that are not well distributed in the image. As a result, the geometry consistency check is robust.

### III. ORTHONORMAL TRANSFORM

Figure 1 plots the covariance matrix of the 64-dimensional feature descriptors for all test query images.



**Fig. 1.** Covariance matrix of the 64 dimension feature descriptors for all query images

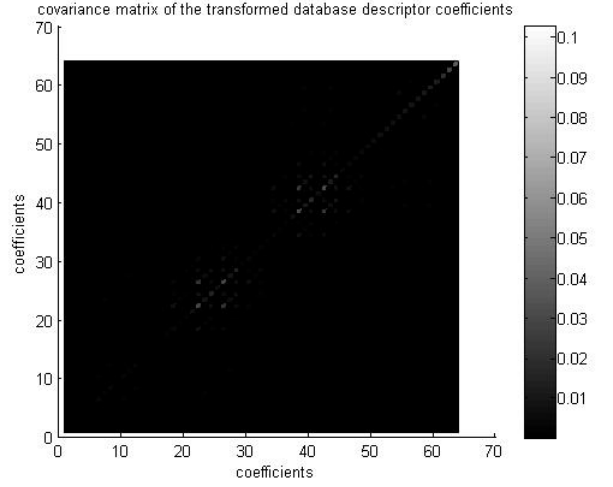
Encoding the samples in another domain can improve the compression rate-distortion curve. We explored two well-known transformation matrices used in image compression: KLT and DCT.

#### A. Karhunen-Loève Transform

The Karhunen-Loève Transform (KLT) is the most optimal orthonormal transformation to use because it is derived from the sample statistics; the columns of the KLT matrix,  $K$ , correspond to the eigenvectors of  $C_X$ . For image transformation, KLT is not practical because it cannot apply to different image blocks. This is not an issue for feature descriptors. We obtain the KLT matrix using the covariance matrix of a set of feature descriptors corresponding to some database image.

$$C_{Y,database} = K_{database}^{-1} * C_{X,database} * K_{database}$$

Figure 2 shows  $C_{Y,database}$ ; note that many transform coefficients are now independent to each other (their covariance is 0). The KLT have localized the energy of the coefficients.

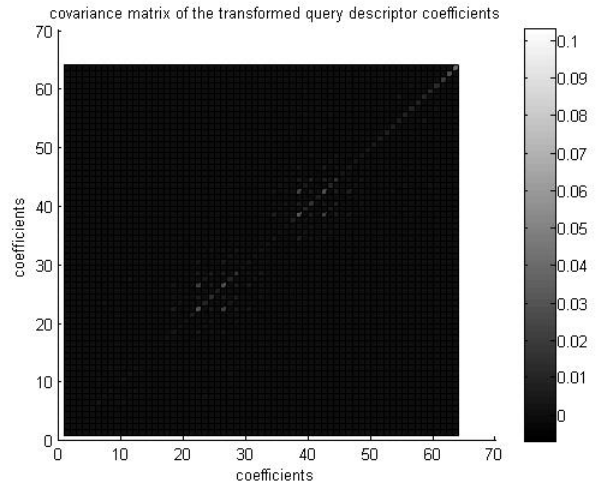


**Fig. 2.** Covariance matrix of the transformed feature descriptors corresponding to the database set.

KLT can be impractical because we have to know the sample statistics beforehand. However, given a good set of query images whose features has a good match rate to the features found in the database images, we can show that it is only necessary to know the statistics of the database features and not the statistics of the query features.

$$C_{Y,query} = K_{database}^{-1} * C_{X,query} * K_{database}$$

Because the KLT matrix is not specifically tailored to the sample data, many of the transform coefficients are no longer independent from each other.



**Fig. 3.** Covariance matrix of the transformed feature descriptors corresponding to the query set.

We can also compare the performance of using a tailored KLT matrix or an approximate one by looking that the coding gain of  $Y_{database}$  and  $Y_{query}$ . The coding gain is:

$$\frac{d(R)}{d^{XFORM}(R)} = \frac{\left(\frac{1}{N}\right) \sum_{n=0}^{N-1} \sigma_{Y_n}^2}{\sqrt{\prod_{n=0}^{N-1} \sigma_{Y_n}^2}}$$

for an orthonormal transform [3].

We find that the coding gain for using  $K_{database}$  to be 3.9971 for  $Y_{database}$  and 3.129 for  $Y_{query}$ . There is little performance degradation in encoding SURF feature descriptors using KLT matrix derived using statistics of “matched” feature descriptors from a different data set.

### B. Discrete Cosine Transform

We have shown that an approximate KLT sufficient for encoding feature descriptors provided that the KLT is derived from matched features descriptors. However, this method still relies on sample statistics of some sort.

In image compression, the Discrete Cosine Transform (DCT) offers comparable performance as the KLT with the additional advantage that the DCT does not require information about the sample statistics. We explored the usage a 1x64 DCT for encoding feature descriptors.

Fig. 4 shows the covariance matrix of the DCT transformed coefficients. We can see that energy concentration is not as good as the KLT. DCT transformation only has a coding gain of 1.31. The DCT is not a good approximation for the KLT for feature descriptors.

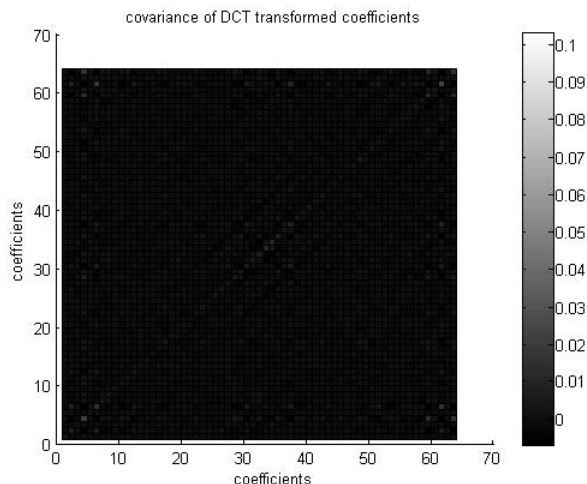


Fig. 4. Covariance matrix of the DCT transformed coefficients.

As a compromise between performance and practicability, we used the KLT matrix derived from the database features to determine the transform coefficients of the query features.

## IV. QUANTIZATION AND ENTROPY CODING

The transformed feature descriptors would take values ranging from -1 to 1. In this project, we use mid-tread uniform quantization with the same step-size applied across all coefficients, which is shown in [3] to perform best to minimize mean-squared-error (MSE) distortion. The quantization level is varied between 1 bit (corresponds to step size of 2) and 11-bit uniform mid-tread quantization (corresponds to step size of 0.000977).

Because the transformed coefficients are not statistically independent to each, the entropy bound we calculate on the data, is greater than the Shannon Lower Bound of the data.

$$SLB = H(X) < H(X_{coeff1}) + \dots + H(X_{coeff64})$$

All entropy coding methods requires knowledge of the PMF of the symbols to be encoded. From our exercise involving the KLT, we have found that matched database and query images share similar feature statistics. Therefore, all the statistics that we use to encode the query feature descriptors were derived from feature descriptors of the database images.

Since the covariance between the different transform coefficients are small, we used a different VLC codebook for each of the 64 transform coefficient symbols; this necessitate us to precompute 64 different PMFs before encoding begins. As the different transform coefficients are not statistically independent to each other, we understand that this will introduce some inefficiency into our system.

### A. Huffman coding

Given a finite alphabet and the corresponding probability measure function (PMF), Huffman coding seeks to minimize the average codeword length of the entire alphabet; more likely symbols are given shorter codeword and less likely symbols are given longer codeword. Codeword are assigned in a tree like structure as shown in Fig. 5.

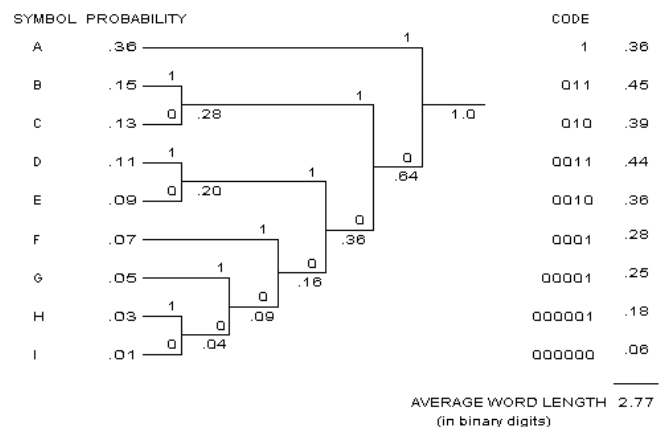


Fig. 5. Huffman coding structure [4].

In our experiment, we found that many transform coefficients, particularly those with low energy and subjected to coarse quantization steps, tend to be quantized to very few bin values, resulting in a long run of the same value. It is wasteful to devote a separate codeword (even if it is just 1 bit) to each

of these symbols. Therefore, we used modified Huffman with run-length encoding next to try to reduce our bit rate.

### B. Modified Huffman with run-length encoding

We use the same Huffman codebooks as in the previous part. However, when the encoder encounters a symbol that has probability of occurrence greater than 0.7, the encoder switches to run-length mode. In run-length mode, the encoder keeps track of the length of the symbols run, when it encounters a different symbol, the encoder transmits only the symbol value and the length of the run.

### C. Arithmetic Coding

Arithmetic coding is a finite precision implementation of the Elias coding, which incrementally constructs a single codeword for an arbitrarily long sequence of source symbols as they arrive [3]. We extend the binary arithmetic coder to code K-bit integer data. The sufficiency of binary arithmetic coder to code  $2^K$  symbols (K-bit integer) is discussed in [3] and [6]. Given a working binary arithmetic coder, the task then is simply to provide the conditional symbol probabilities, which could be derived from the PMF of the coefficients.

## V. RESULTS

Figure 6 plots the rate-distortion curve of transformed feature descriptors entropy-coded using Huffman coding, modified Huffman coding with run-length, and arithmetic coding. The distortion measure is the sum of squared error per descriptor.

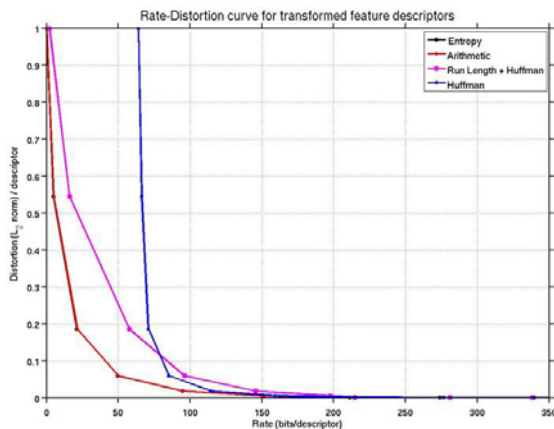


Fig. 6. Rate distortion curve of different entropy encoding method applied to transformed coefficients.

As expected, the performance of Huffman coding (blue curve) is particularly worse at low rate. This could be improved by the using modified Huffman with run-length. Not surprisingly, the binary arithmetic encoder performed the best and matches with the minimum entropy measurement of the system:

$$\text{entropy} = H(X_{\text{coeff}1}) + H(X_{\text{coeff}2}) + \dots + H(X_{\text{coeff}64})$$

This entropy however, is larger than the Shannon Lower Bound of the system as the transform coefficients are not statistically independent.

Figure 7 plots the query-to-database image matching rate as a function of the sum of squared difference per descriptor.

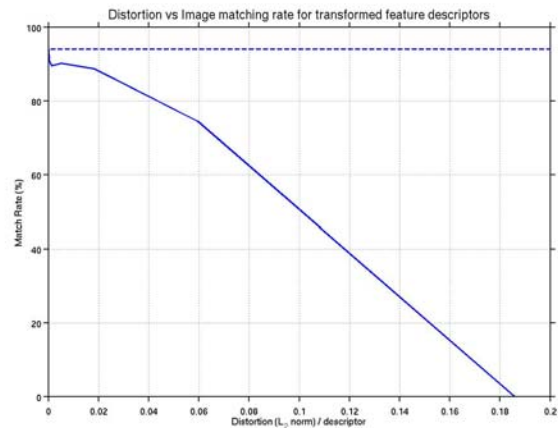


Fig. 7. Plot of MSE distortion vs. feature descriptor match rate. Dotted line denotes the baseline, which is the match rate between database features and uncompressed query features.

Figure 7 shows that the match rate between the reconstructed query features and the stored database features drop to 0% if MSE distortion rate between query features and reconstructed query features is larger than 0.13. In fact small changes in MSE distortion will have some effect on the match rate as the feature matching algorithm employed by SURF is not particularly robust against distortions to the feature descriptors. This is possibly due to the low dimensionality of the SURF descriptors. However, an acceptable match rate ( $\approx 87\%$ ) can be achieved if the distortion is approximately 0.02 or less. Figure 8 shows the close-up of the Figure 6 in the region.

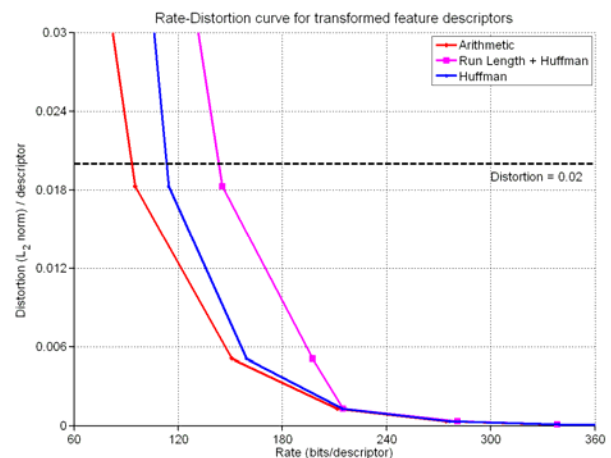


Fig. 8. Rate distortion curve of Huffman and arithmetic coding near the threshold of distortion = 0.02

The distortion of 0.02 roughly corresponds to 5-bit quantization level. Figure 8 shows that this corresponds to bit rate of 96 bits/descriptor for arithmetic coding, 115 bits/descriptor for Huffman coding, and 146 bits/descriptor for

modified Huffman coding with run-length. Around this rate, the performance of arithmetic coding still outperforms both Huffman and modified Huffman with run-length. It is also noteworthy that at this level, modified Huffman with run-length actually performs worse compared to Huffman coding. The three entropy coding schemes converge starting from 7-bit quantization level, which corresponds to a bit rate of 210 bits per descriptor and distortion value of 0.0013.

## VI. CONCLUSION

Contrary to regular image compression, KLT is a viable and practical transform matrix for compression SURF feature descriptors; the statistics of some known database set who matches well to the image features that we are interested in compressing will suffice for successful implementation of KLT transform coding and lossless entropy encoding. This is a welcome conclusion because it means that for successful feature matching, the query image does not need to be sent, only the feature descriptors found from the query image.

We found that in order to achieve an acceptable match rate of 87%, the sum of squared error per descriptor has to be less than 0.02. The minimum bit rate to achieve this goal is 96 bits/descriptor using arithmetic coding. Overall, we found that arithmetic coding works best to compress the transformed feature vectors.

One disadvantage of the method that we process is that the final match rate between the features is not very robust against distortion. A slight increase in distortion will cause a dramatic decrease in match rate. It would be interesting to see if this behavior extends to other feature detection/matching algorithms as well or if it is a characteristic of SURF descriptors.

The next step would be to characterize how well KLT transform coding use transform matrices and statistics generated from a less well matched database-query feature set would perform.

## ACKNOWLEDGMENT

The code we used to generate the actual Huffman code table is found at [2]. We would like thank David Chen, Vijay Chandrasekhar, Professor Girod, Gabriel Takacs, and David Varodayan for their support and help, especially to help us understand the conceptual differences between image compression and feature compression.

## REFERENCES

- [1] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: Speeded Up Robust Features" in *ECCV* (1), 2006, pp. 404-417.
- [2] G. Takacs, et al., "Outdoors Augmented Reality on Mobile Phone using Loxel-Based Visual Feature Organization," submitted to *IEEE Trans. Pattern Analysis and Machine Intelligence*, available online at [http://mars0.stanford.edu/wiki/index.php/Main\\_Page](http://mars0.stanford.edu/wiki/index.php/Main_Page).
- [3] D. S. Taubman and M. W. Marcellin, "JPEG2000: Image Compression Fundamentals, Standards and Practice", Kluwer Academic Publishers, 2002.
- [4] N.Mikael, Available: <http://mikaeli.mikkeli.amk.fi/mikaeli/info/tutkimukset/wang/index.htm>

- [5] ECE/CS533 Matlab M-files. Available: <http://homepages.cae.wisc.edu/~ece533/matlab/index.html>
- [6] B. Girod, *Lecture Notes for EE 398: Image and Video Coding*, Winter 2008.

## APPENDIX

**Ivan Janatra** is currently a Masters student in Electrical Engineering at Stanford University. For this project, he investigated several approaches for quantization and coded the K-bit integer arithmetic coding.

**June Zhang** is currently a Masters student in EE at Stanford. For this project, she investigated orthogonal transformation matrices such as the KLT, DCT, and the Haar wavelet transform. She also coded Huffman coding and modified Huffman with run-length coding.