Location Coding for Mobile Image Retrieval

Sam S. Tsai*, David Chen*, Gabriel Takacs*, Vijay Chandrasekhar*,
Jatinder P. Singh† and Bernd Girod*

*Information Systems Laboratory, Stanford University, Stanford, CA 94305, U.S.A.
†Deutsche Telekom Inc. R&D Lab, Los Altos, CA 94022, U.S.A.

*{sstsai, dmchen, gtakacs, vijayc, bgirod}@stanford.edu, †jatinder.singh@telekom.de

ABSTRACT

For mobile image retrieval, efficient data transmission can be achieved by sending only the query features. Each query feature is composed of a descriptor and a location in the image. The former is used to find candidate matching images using a "bag-of-words" approach while the latter is used in a geometric consistency check to map features in the query image to corresponding features in the database image.

We investigate how to compress the location information and how lossy compression affects the geometric consistency check. The location information is converted into a location histogram and a context-based arithmetic coding with location refinement method is then proposed to code the histogram. The effects of lossily compressing the location information are evaluated empirically in terms of the errors in corresponding features and the error of the estimated geometric transformation model. From our experiments, rates at ∼5.1 bits per feature can achieve errors comparable to lossless coding. The proposed scheme achieves a 12.5× rate reduction compared to the floating point representation, and 2.8× rate reduction compared to a fixed point representation.

Categories and Subject Descriptors: H.3 [Information Storage and Retrieval]: Miscellaneous

General Terms: Algorithm

Keywords: scale-invariant feature, scalable vocabulary tree, mobile augmented reality, content-based image retrieval

1. INTRODUCTION

Handheld mobile devices, such as camera-phones or PDAs, are expected to become ubiquitous platforms for visual search and mobile augmented reality applications [1, 2]. For mobile image matching, a visual database is typically stored at a server in the network. Hence, for a visual comparison, information must be either uploaded from the mobile to the server, or downloaded from the server to the mobile. With relatively slow wireless links, the response time of the system critically depends on how much information is transferred in both directions.

1.1 Background on Image Retrieval

Image retrieval based on local features has become an attractive approach [3] since they are robust to lighting, rotation, scale, and mild viewpoint changes. A local feature consists of two pieces of information: the location of the feature and the descriptor of the local region around a feature. Features are formed at interest points in the image that have specific structures such as corners or blobs [4, 5, 6, 7]. These structures are typically invariant to changes in luminance, scale, rotation, and translation. Descriptors are then calculated from the image intensity values in a local region centered at the interest point. A wide range of descriptors were tested in [8] and gradient distribution-based descriptors were shown to perform best. These type of descriptors, such as the Scale Invariant Feature Transform (SIFT) [5], the Gradient Location and Orientation Histogram (GLOH) [8], the Speeded Up Robust Features (SURF) [7], and Compressed Histogram of Gradients (CHOG) [9], are histograms of the image intensity gradients of the local region.

Fast large-scale image retrieval is enabled by using a Scalable Vocabulary Tree (SVT) [11] for matching. Features are extracted from the database of images to form a set of features. A hierarchical k-means clustering algorithm is applied to the set of features, forming the SVT. This method is also known as Tree Structured Vector Quantization (TSVQ). When performing a match in the SVT, descriptors of the query image are quantized through the SVT and a histogram of the node visits on the tree nodes is generated. Candidate images are then sorted according to the similarity of the candidate database image histogram to the query image histogram. The descriptors are treated as a visual "bag-of-words" in the SVT matching.

Geometric consistency check is applied to validate or re-rank the list of candidate images [1, 12]. This is done by pairwise matching the features extracted from a query image to the features extracted from a database image. Typically, the ratio test [5] is used to predict a set of corresponding features between the two images. Then, RANSAC [10] is applied on the predicted set of features to estimated a geometric transformation model and a set of matching correspondences. For image matching applications, a good number of matching correspondences represents a confident match. For augmented reality applications, a good geometric transformation model provides spatially accurate overlaying display. A typical mobile image retrieval system is illustrated in Fig. 1.

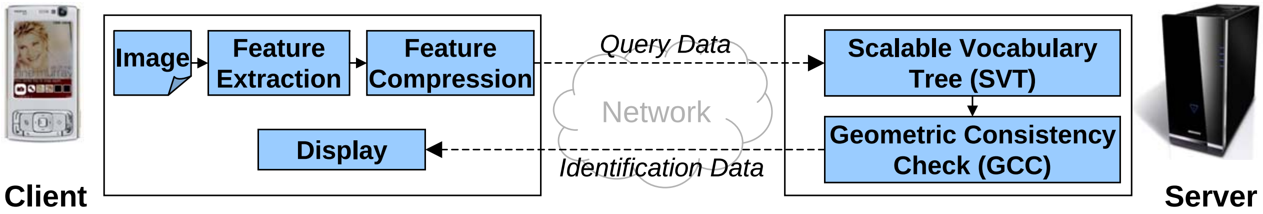


Figure 1: A mobile CD cover recognition system where the server is located at a remote location. Query data is sent over the network and the identification data is sent back to the mobile-phone (from [2]).

1.2 Rate-Efficient Image Retrieval

As mobile phones are equipped with camera devices, they have naturally become an input device for image retrieval systems. However, due to the limited memory and processing power on the mobile phone, large-scale retrieval databases must be stored at a remote location (Fig. 1). In this case, the mobile phone has to send the data over a wireless network to the server. Wireless connections have limited bandwidth, prompting the need to send as little data as possible. In the scenario in which the phone is capable of carrying out feature extraction on the device, only the feature data required for the retrieval system needs to be sent.

Recent work has explored the compression of descriptors. Random projections of SIFT descriptors were proposed to reduce the bit rate [13]. Chandrasekhar et al. studied transform coding of SIFT and SURF descriptors [14]. CHOG, a low bit rate descriptor, was proposed in [9]. CHOG, at the rate of 8 bytes per descriptor, can achieve matching capabilities comparable to the uncompressed features. In [15], the local image patch around the interest point is compressed and sent over the network at low bit rate, which also shifts the workload of descriptor computation to the server side. Unlike the other approaches, the retrieval system in [16] sends a tree histogram in place of individual descriptors, which enables significant additional rate reduction.

1.3 Contributions

A novel method to compress the location information is proposed. The location information is typically of floating point precision, and its size comparable to the size of the descriptors in [9]. The location information is converted into a location histogram and the location histogram is compressed. Empirical evaluation on the effect of lossily compressing the location information is made by examining the errors of the matching correspondences and geometric transformation model estimated by the geometric consistency check. A suitable rate is then chosen for mobile image matching and augmented reality applications.

In Sec. 2, we apply different methods to compress the location histogram, and describe the proposed context-based arithmetic coding with location refinement scheme. Experimental results of the proposed scheme applied in the mobile image retrieval framework and the effects of lossy compression of the location information are presented in Sec. 3.

2. LOCATION CODING

Features are structurally distinguishable points in the image. Ideal distinguishable points are corners or blobs in an image. Because of their distinctive structure, the features are invariant to lighting, scale, and mild viewpoint changes, i.e., they can be reliably detected in different camera views.

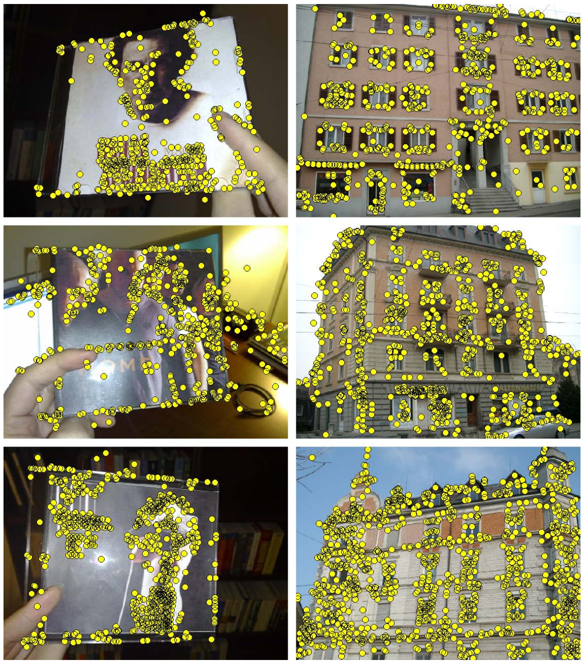


Figure 2: Features in the CD Database (left column) and the Zurich Building Database (right column) are randomly scattered but still cluster near structures.

In SIFT [5], Lowe proposed to detect features by finding extrema points in a pyramid of Difference of Gaussian (DoG) images. A 3D quadratic function is fitted to the local samples in the images to find the sub-pixel location of the interest point. The Hessian of the interest point is then examined to rule out edge responses. Typical DoG interest points within query images are shown in Fig. 2. We use SIFT for our evaluation in this section, and show experimental results of other Hessian-based interest point detectors in Sec. 3.2. In the following section, we describe how to compress the location information in a rate-efficient manner.

2.1 Location Histogram

We aim to reduce the number of bits needed for the location information of each feature. A natural approach to lossily coding the feature locations is to quantize the locations and entropy code the quantized data. In our approach, instead of coding the location information of each feature, we convert the location information of the features into a location histogram, and code the location histogram.

There are two benefits in converting the location information into a location histogram and coding the histogram. First, in

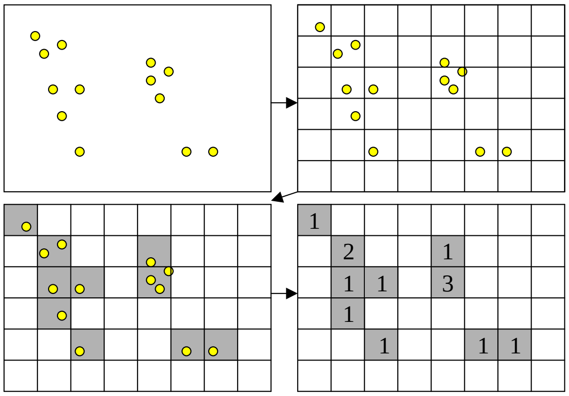


Figure 3: The histogram of the features contains the histogram map and the histogram count.

[16], it was shown that by discarding the order information of the coded items the bit rate can be reduced. Second, the interest points are structural points in the image, and hence, spatial structure relationship can be exploited when they are coded.

2.2 Compression of Location Histogram

The location histogram is represented as the histogram map and histogram count, Fig. 3. The histogram map is formed by the map of empty and non-empty histogram blocks, while the histogram count is the number of features in the non-empty block. Different sizes of histogram block yield different quantization levels on the location information. The histogram map and histogram count is coded separately. Spatial relationship of the interest points is exploited when coding the histogram map.

Histogram Map Coding. The histogram map resembles a binary image with '1' indicating a non-empty block and '0' otherwise. To find a suitable method for coding this type of data, we experiment with the following methods to code the histogram map to find which method performs the best:

- *Runlength*. The empty blocks in the histogram map is converted into runlengths in raster scan order. The runlengths are then coded using an entropy coder.
- *Quad-tree*. The histogram map is converted into a quad-tree [17]; the map is divided into four squares, and each square containing non-empty blocks is divided into four squares recursively to form a tree. A binary representation of the tree is generated by traversing the tree breadth first using symbols that indicate whether the square has children or not. The binary representation is entropy-coded with different entropy codes for different levels of the tree.
- *Context-based*. The histogram map is entropy-coded block by block in the raster scan order with similar context information used in coding binary images [18]. The neighboring blocks of the current coded block is used as context, Fig. 5. Here, we use 14 nearest available pixels to form the context.

We examine the coding performance of the different coding methods described above by compressing DoG interest points extracted from the Zurich Building Database (ZuBuD) and CD Database (CDD) images, which we will describe in Sec. 3.

Table 1: Entropy of the Histogram Map

	32×32	16×16	8×8	4×4	2×2
Bits/Image	222	807	2014	3730	5552
Bits/Feature	0.247	0.900	2.246	4.160	6.192

Table 2: Entropy of the Histogram Count

	32×32	16×16	8×8	4×4	2×2
Bits/Feature	0.699	0.967	0.948	0.766	0.708

In Fig. 4, we plot the entropy for different histogram block sizes of the three coding methods. We find that Context-based coding method performs the best among the three. However, the gains of the Context-based coding method decreases as the block size becomes smaller. Runlength coding for larger histogram block size does not provide any gain. Based on these results, Context-based coding method is most suitable for coding histogram maps. We show the entropy of the proposed method in bits per feature and bits per image in Table 1, with an average of 900 features per image.

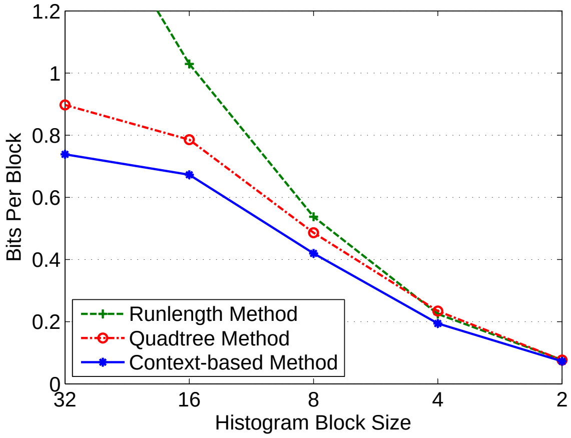


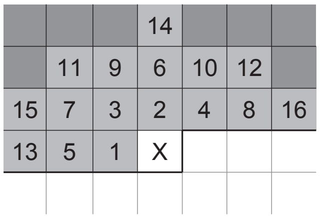
Figure 4: Rate of three coding methods on the histogram map.

Histogram Count Coding. The histogram count is coded separately from the histogram map. We show the entropy of the histogram count in bits per feature for the ZuBuD and CDD datasets in Table 2. For histogram block sizes smaller than 8×8, the rate of the histogram count is much smaller than the rate of the histogram map.

2.3 Context-based Arithmetic Coding with Location Refinement

Context-based coding provides the best compression performance as shown in the previous section. In this section, we further examine how the context size affects the coding performance, and will describe the two stage context-based arithmetic coding with location refinement scheme that we propose.

Different Context Size. We examine how larger context sizes can affect the coding performance. In Fig. 5, a map labeling the neighboring pixels in the order of the distance to the coded pixel is shown. We increase the context size by adding more neighboring pixels in the context, with nearest first. Context-k denotes a context consisting of k nearest neighboring pixels. We code the histogram map using no context and also code the histogram map using different sizes of context. The entropy of using different contexts shown in Fig. 6. The coding performance improves as the context size increases, but the improvement saturates when the number of pixels in the context grows beyond 16. This is due to context dilution, i.e., the context size has grown too large for effective training. In our experiments, we use Context-14 as the context model.



			14			
	11	9	6	10	12	
15	7	3	2	4	8	16
13	5	1	X			

Figure 5: When coding pixel 'X' the neighboring pixels are included in the context. Context-8 denotes a context map consisting of neighboring pixels $1, 2, 3, 4, 5, 6, 7, 8$.

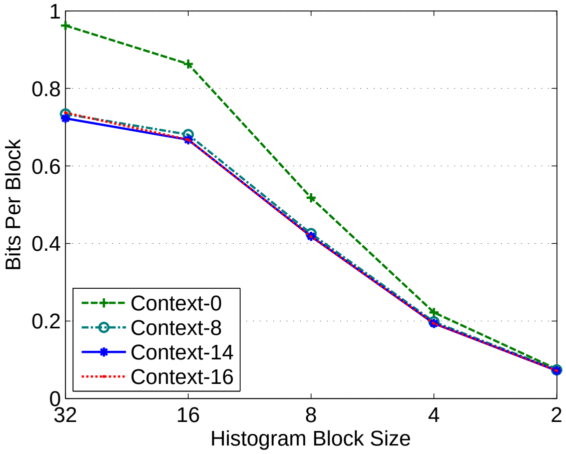


Figure 6: Rate of different sizes of context on the histogram map.

Context-based Arithmetic Coding with Location Refinement (CBAC-LR). Structures in the image vary little in local regions. The distance between interest points usually is greater than 4 pixels. Thus, the benefits of using context-based coding on histogram maps with histogram block size smaller than 4×4 is smaller. In Fig. 4, better coding performance of the context-based coding method is observed for only histogram block sizes of 4×4 and larger. Thus, we propose another approach for coding smaller histogram block sizes.

The proposed two-stage coding method of the location histogram is as follow. To code the location information in a histogram with histogram block size greater than 2×2, the context-based binary arithmetic coder is applied to code the histogram map, and a multi-symbol arithmetic coder is used to code the histogram count. To code the location information in a histogram with histogram block size smaller than 4×4, we code the location histogram with histogram block size 4×4, and use refinement bits to specify the vertical and horizontal location of each feature in the histogram block. For example, an additional 2, 4 refinement bits are used for 2×2 blocks and 1×1 blocks respectively for each feature.

3. EXPERIMENTAL RESULTS

We apply the proposed scheme to compress query features in a mobile image retrieval framework and empirically evaluate the results. We assume a matching candidate image to a query image is found from a large database using SVT histogram matching. Geometric consistency check (GCC) is applied to validate the match between the candidate image and the query image.

In our experiments, two datasets are used. The first is the widely used Zurich Building Database[1], which consists of different views of 201 different building facades in Zurich. The second is our CD Database[2]. 50 CD covers are randomly chosen from a CD image database of 10,597 entries, and photographed in different angles and backgrounds. For our experiments, we resize the query images to 640×480 resolution.

The geometric consistency check is described in detail in Sec. 3.1. In Sec. 3.2, we show the results on how the rate trades off with the loss of matching correspondences after the geometric consistency check. In Sec. 3.3, we empirically examine how lossy compression of the location information causes errors in the GCC.

3.1 Geometric Consistency Check

GCC is used to validate the match between the query image and the candidate image. The check consists of two steps: 1) the ratio test [5], and 2) RANSAC [10].

We describe the process using the following notations. For a query image I_Q, a set of features $F_Q = \{l_{q,i}, d_{q,i}\}_{i \in 1,2,\ldots N_Q}$ is extracted. l denotes the location, d denotes the descriptor, and N_Q denotes the total number of features. We match the query image to a candidate database image I_C, where a set of features $F_C = \{l_{c,i}, d_{c,i}\}$ is extracted with a total of N_C number of features.

The ratio test establishes a set of correspondence $\hat{C}_{RT} = \{\hat{m}_{q,j}, \hat{m}_{c,j}\}_{j \in 1,2,\ldots \hat{N}_{RT}}$ given F_Q and F_C. $\hat{m}_{q,j}$, $\hat{m}_{c,j}$ are indices to the features in F_Q, F_C respectively and indicate matching features. $\hat{N}_{RT}$ is the total number of estimated correspondences. In our experiments, we use 0.8 as the ratio threshold in the ratio test.

RANSAC is applied to the set of correspondences, $\hat{C}_{RT}$, to find a geometric transformation, $\hat{M}_{GCC}$. We can transform the database feature locations in $\hat{C}_{RT}$ using $\hat{M}_{GCC}$. The transformed location is noted as $T_{\hat{M}_{GCC}}(l_{c,\hat{m}_{c,j}})$, where $T_M(l)$ is the new image location of the location l using M as the geometric transformation model. The *location error* of each pair in $\hat{C}_{RT}$ is given as $\|T_{\hat{M}_{GCC}}(l_{c,\hat{m}_{c,j}}) - l_{q,\hat{m}_{q,j}}\|_2$. We say the pair agrees with the model $\hat{M}_{GCC}$ if its *location error* is smaller than Θ_l. The set of correspondences that agrees with the model is denoted as $\hat{C}_{GCC}$.

3.2 Compression of Location Information

Compression of the location information of query features is carried out using our scheme. As the rate allocated for location information is decreased, the distortion in the location information increases. *Location error* of each pair in $\hat{C}_{RT}$ increases, and thus reducing the number of correspondences in $\hat{C}_{GCC}$. We define *feature loss percentage* as the

[1] http://www.vision.ee.ethz.ch/showroom/zubud/

[2] http://msw3.stanford.edu/∼dchen/CDD/index.html

percentage of losses in the number of correspondences when compared to a scheme without compression. This measure is typically used when no ground truth data is available.

We show the feature loss percentage of two coding schemes: *CBAC-LR* and *fixed-length coding*. The former is the proposed scheme described in Sec. 2.3 and the latter is a coding scheme where the horizontal and vertical coordinates of each feature is coded in a fixed number of bits separately.

In addition, we also present two other results: *entropy* and *histogram*. *Entropy* is the entropy of the *CBAC-LR*. *Histogram* is the theoretical rate achieved by representing the location information in a location histogram. The location histogram is an orderless representation in which the information of order is discarded. Therefore, there is a potential bit rate reduction of $log_2(N!)$ [16], where N is the total number of features in the image. The average number of features per query image in the data set is 750. Using the Stirling's approximation to calculate $log_2(N!)$, we find that a theoretical rate reduction of ~8.1 bits per feature over *fixed-length coding* can be achieved.

In Fig. 7, we show the results *feature loss percentage*. Although the training set of the Context-based coding method is different from the test set, the rate is close to entropy, suggesting that the context model is applicable across different images. The proposed method provides a ~9 bit rate reduction per feature when compared with *fixed-length coding* method and provides a ~1 bit rate reduction per feature over the theoretical rate shown as *histogram*.

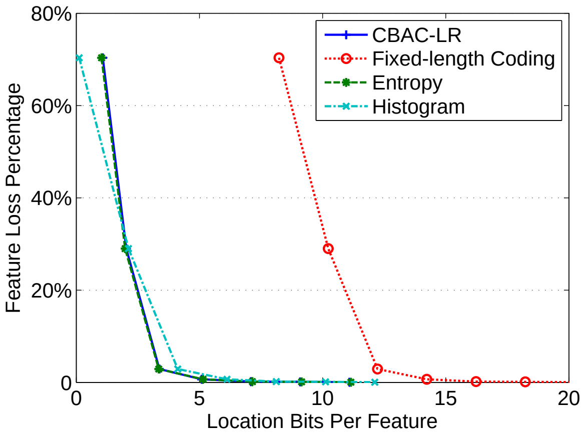


Figure 7: Feature loss percentage vs. rate of the proposed scheme

Compression of Hessian-based Detectors. We experiment with two other types of detectors, the Hessian-Laplace detector [6] and the SURF Fast Hessian detector [7], and show the results in Fig. 8. The parameters of the detectors were tuned to give an average of 750 feature counts per image. For fair comparison, we examine the results using the location quantization error with the purpose to rule out the influences of descriptors.

3.3 Effect of Lossy Compression of Location

In this section, we examine how lossily compressing the location information will affect the GCC. To simplify the presentation, we describe the levels of lossy compression using different histogram block sizes.

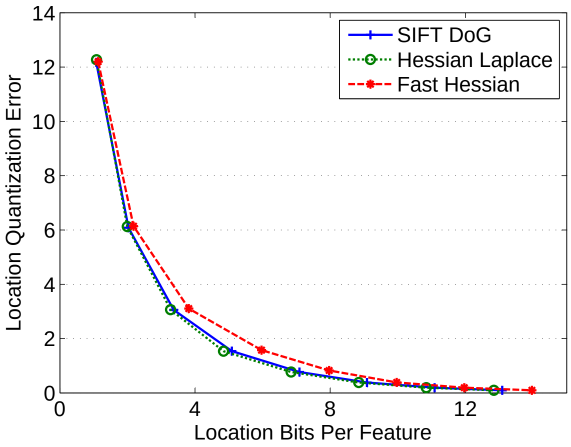


Figure 8: Location quantization error vs. rate for different interest point detectors

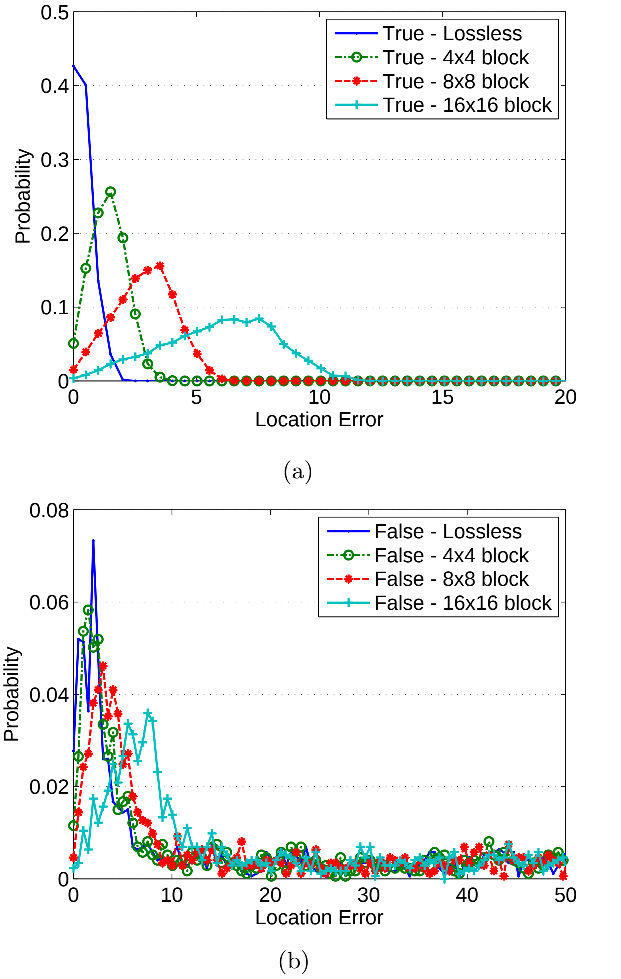
We first create a set of ground truth data for the GCC using the CD dataset. We first manually find a homography M_{GT} between the database image and the query image. Then, ground truth correspondences are established if they satisfy the following criteria: 1) the projected database image feature patch and the query image feature patch have overlap error [8] smaller then 40%, 2) the location error between projected database feature location and the query feature location is within 2 pixels, and 3) the orientation difference of the projected database feature and the query feature is smaller than 45°. The set of ground truth correspondences is denoted as C_{GT}.

To quantify the comparison, we use two error measurements: 1) correspondence error, and 2) geometric transformation model error. The first is the error of matching correspondences between the pair of images. The geometric transformation model error depicts the displacement in augmented displays to the query image.

3.3.1 Correspondence Error

The correspondence error is the error of $\hat{C}_{GCC}$ when compared with the ground truth correspondences, C_{GT}. For each pair of correspondence in C_{RT}, it is labeled as *true* if it is in C_{GT} and *false* if otherwise. We plot the *location error* distribution of *true* and *false* matches in Fig. 9, while using M_{GT} to project the true location. When larger histogram block sizes are used, the error distance of *true* matches increase. *False* matches are uniformly distributed over the image and the distribution changes not much due to compression. The peak in small location errors for *false* matches corresponds to interest points at the wrong scale or orientation.

The set $\hat{C}_{GCC}$ is determined by the geometric transformation model and the given error threshold Θ_l. We assume that the estimated model is found to be M_{GT}. Then, by varying the threshold Θ_l, we plot the receiver operating characteristic curve, Fig. 10. The equal error rate curve representing equal false positive and false negative probabilities is shown in Fig. 11. To operate at the equal error rate curve, larger Θ_l is used for larger histogram block sizes. An inherent error of 10% is observed for GCC. For histogram block size 4×4 and 4×4, the additional error is 3% and 8% compared with the lossless case.



(a)

(b)

Figure 9: Distribution of location error for (a) *true* matches, and (b) *false* matches for different histogram block sizes.

3.3.2 Geometric Transformation Model Error

The geometric transformation model error is the deviation of the estimated model, found by GCC, to the true geometric transformation. Here, we use a homography. To evaluate the estimated homography $\hat{M}_{GCC}$, we compare it with the ground truth homography M_{GT} using the following measure:

$$Error(\hat{M}_{GCC}) = \frac{1}{N_C} \sum_{i=1}^{N_C} \|T_{\hat{M}_{GCC}}(l_i) - T_{M_{GT}}(l_i)\|_2. \quad (1)$$

Examples of projecting items using the estimated homography model are shown in Fig. 12. Different quantization level is applied to the location information, resulting in the different estimated models.

For a matching query feature set and database feature set, we run the GCC for 100 times and average the errors. The estimated model error for four different image pairs is shown in Fig. 13. Each pair corresponds to a different number of corresponding matches. The error is correlated with the number of corresponding matches. As more observations on the unknown model is made, the error becomes smaller. The error of the model is approximately the same for histogram block sizes smaller than 4×4.

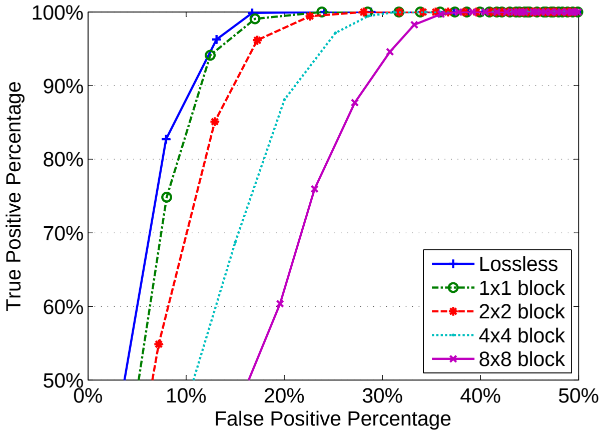


Figure 10: Receiver operating characteristic curve for different histogram block sizes.

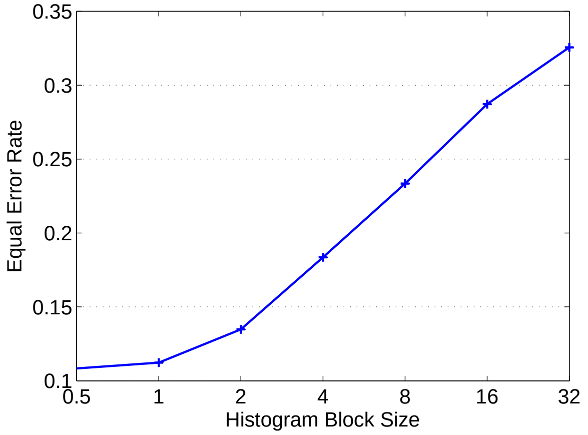


Figure 11: Equal error rate operating curve for different histogram block sizes.

4. CONCLUSIONS

We have proposed a novel method of compressing location information for mobile image retrieval systems based on feature-based matching. Location information is converted into a location histogram and the location histogram is compressed. Of the three different schemes examined, context-based coding of the location histogram performed the best. It is further improved by introducing a two-stage coding method using refinement bits. We empirically evaluated the effects of lossily compressing location information on geometric consistency check. Errors is quantified using the correspondence error and geometric transformation model error.

From our experimental results, it is shown that by coding the location information in 4×4 histogram blocks, the error of the model is smaller than 1 pixel and correspondence match is within 90% error of the lossless compression case. When coding the location information using the 4×4 histogram block size, the rate is ~5.1 bits per feature. Thus, we have achieve 12.5× rate reduction compared with the floating point representation, and 2.8× rate reduction compared with a fixed point coding.

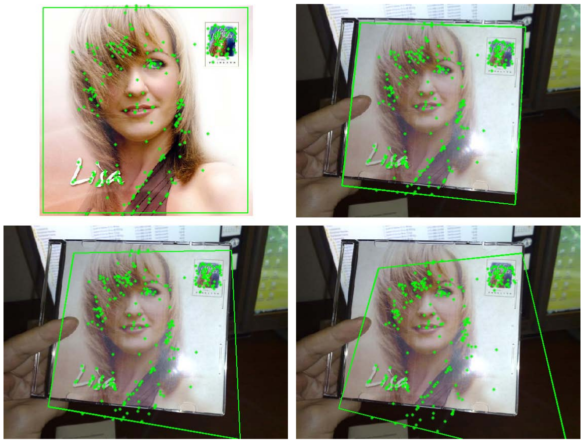


Figure 12: Top-left: database image with features and an overlay box. Top-right: query image with projected features and overlay box using a homography model estimated from compressed feature locations using histogram block size 4×4. Bottom-left: histogram block size 8×8. Bottom-right: histogram block size 16×16.

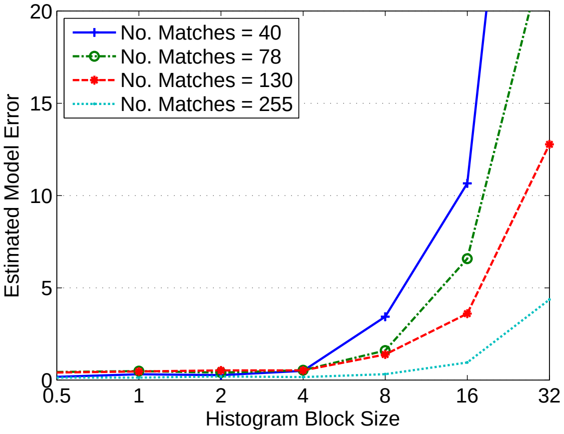


Figure 13: Estimated model error of four different images with different number of matching features.

5. REFERENCES

[1] G. Takacs, V. Chandrasekhar, N. Gelfand, Y. Xiong, W. Chen, T. Bismpigiannis, R. Grzeszczuk, K. Pulli, and B. Girod, "Outdoors augmented reality on mobile phone using loxel-based visual feature organization," in *ACM International Conference on Multimedia Information Retrieval*, Vancouver, Canada, October 2008.

[2] S. S. Tsai, D. Chen, J. Singh, and B. Girod, "Rate-efficient, real-time CD cover recognition on a camera-phone," in *ACM International Conference on Multimedia*, Vancouver, Canada, October 2008.

[3] C. Schmid and R. Mohr, "Local greyvalue invariants for image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 530–535, 1997.

[4] T. Lindeberg, "Feature detection with automatic scale selection," *International Journal of Computer Vision*, vol. 30, no. 2, pp. 79–116, November 1998.

[5] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, November 2004.

[6] K. Mikolajczyk and C. Schmid, "Scale and affine invariant interest point detectors," *International Journal of Computer Vision*, vol. 60, no. 1, pp. 63–86, October 2004.

[7] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: speeded up robust features," in *European Conference on Computer Vision*, Graz, Austria, May 2006, pp. 404–417.

[8] K. Mikolajczyk and C. Schmid, "Performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, October 2005.

[9] V. Chandrasekhar, G. Takacs, D. Chen, S. S. Tsai, R. Grzeszczuk, and B. Girod, "CHoG: Compressed Histogram of Gradients," in *submitted to Conference on Computer Vision and Pattern Recognition*, 2009.

[10] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cryptography," *Communications of ACM*, vol. 24, no. 1, pp. 381–395, 1981.

[11] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Conference on Computer Vision and Pattern Recognition*, New York, NY, USA, June 2006, pp. 2161–2168.

[12] J. Philbin, O. Chum, M Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[13] C. Yeo, P. Ahammad, and K. Ramchandran, "Rate-efficient visual correspondences using random projections," in *International Conference on Image Processing*, San Diego, CA, USA, October 2008.

[14] V. Chandrasekhar, G. Takacs, D. Chen, S. S. Tsai, J. Singh, and B. Girod, "Transform coding of feature descriptors," in *SPIE Visual Communications and Image Processing*, San Jose, CA, USA, January 2009.

[15] M. Makar, C.-L. Chang, D. Chen, S. S. Tsai, and B. Girod, "Compression of image patches for local features extraction," in *International Conference on Acoustics, Speech, and Signal Processing*, Curitba, Brazil, April 2009.

[16] D.M. Chen, S. S. Tsai, V. Chandrasekhar, G. Takacs, J. Singh, and B. Girod, "Tree histogram coding for mobile image matching," in *Data Compression Conference*, Snowbird, UT, USA, March 2009.

[17] G.J. Sullivan and R.L. Baker, "Efficient quadtree coding of images and video," *IEEE Transactions Image Processing*, vol. 3, no. 3, pp. 327–331, May 1994.

[18] International Organization for Standardization, "Progressive bi-level image compression," ISO/IEC International Standard 11544, ITU Recommendation T.82, 1993.