KEYPOINT ENCODING AND TRANSMISSION FOR IMPROVED FEATURE EXTRACTION FROM COMPRESSED IMAGES

Jianshu Chao[⋆] *Eckehard Steinbach*[⋆] *Lexing Xie*[†]

[⋆]Technische Universität München, Germany; [†]Australian National University, Australia
[⋆]{jianshu.chao, eckehard.steinbach}@tum.de; [†]lexing.xie@anu.edu.au

ABSTRACT

In many mobile visual analysis scenarios, compressed images are transmitted over a communication network for analysis at a server. Often, the processing at the server includes some form of feature extraction and matching. Image compression has been shown to have an adverse effect on feature matching performance. To address this issue, we propose to signal the feature keypoints as side information to the server, and extract only the feature descriptors from the compressed images. To this end, we propose an approach to efficiently encode the locations, scales, and orientations of keypoints extracted from the original image. Furthermore, we propose a new approach for selecting relevant yet fragile keypoints as side information for the image, thus further reducing the data volume. We evaluate the performance of our approach using the Stanford mobile augmented reality dataset. Results show that the feature matching performance is significantly improved for images at low bitrate.

Index Terms— mobile visual search, feature-preserving image compression, feature extraction, feature compression

1. INTRODUCTION

Feature extraction is a fundamental step in many computer vision tasks. Typically, first keypoints (also called interest points or salient points) are detected in images using a detector algorithm; second, distinctive feature vectors (called descriptors) are calculated from image patches located around the keypoints. The feature extraction process ideally identifies features that are shift-invariant, scale-invariant, rotation-invariant, illumination-invariant, etc. Matching features between two images have similar descriptors.

In some applications such as image retrieval, mobile visual search, and object recognition, it is sometimes preferable to run the computer vision algorithms on a server rather than locally on a mobile device. To this end, either compressed images are transmitted to the server and feature detection and descriptor calculation are performed at the server, or features are extracted on the mobile device and compressed versions thereof are uploaded. In the case of uploading compressed images, it is important that the compression impact on feature extraction is as small as possible. Ideally, features extracted from the compressed images are identical or at least very similar to features that are extracted from the uncompressed version. In the case of uploading compressed features, the compression of the keypoint information (location, scale, orientation) and the descriptor should also lead to a small distortion compared with the uncompressed features. Both approaches fall into the emerging area of feature-preserving compression approaches. Previous work on feature-preserving compression can be categorized into three approaches. The first approach is to directly compress the features. The authors in [1–5] propose various approaches to encode Scale Invariant Feature Transform (SIFT) [6] features and binary features extracted from images or video sequences. The emerging Compact Descriptors for Visual Search (CDVS) [7] standard aims to standardize the technologies for compressing features at low bitrate. The second approach is to compress canonical image patches and transmit or store these patches for further processing, e.g., [3, 8–10]. The third approach is to modify standard image compression algorithms in such a way that the features extracted from the compressed images are as similar as possible to the features extracted from the uncompressed image. To this end, [11, 12] optimize the quantization table of JPEG, while [13] modifies the rate allocation to preserve the most important features. Compared with the first two approaches, the merit of feature-preserving image compression is that the decoded images can be viewed and stored for future use, and other types of features can also be extracted. For standard-compatible solutions such as [11–13], standard image decoders can be used to decode the image.

We consider a scenario where images captured by a mobile device are encoded using JPEG, H.264/AVC Intra mode, or H.265/HEVC Intra mode. The compressed images are transmitted via a wireless network to a server where features are extracted. In our previous work [14], we have observed that for feature-preserving image/video compression, the keypoint detection is easily affected by compression artifacts, while the descriptor calculation is more robust. To address this issue, we investigate in this paper whether explicit signaling of keypoint information (location, scale, and orientation) improves the quality of features which are extracted from the compressed images. In the literature, there is some re-

lated work on keypoint encoding. [15] and CDVS [7] propose approaches to encode the locations of keypoints using sum-based context-based arithmetic coding. [16] and [17] propose to encode the locations, scales, and orientations for features with large scales. They encode the downsampled image with a fixed QP, and only a few keypoints with large scales are preserved, because the small scale features are of no use in their scenario. Then, the downsampled image and the encoded keypoints are transmitted to the server. In this paper, we are interested in the feature preservation performance for features with different scales at low bitrate. We build on top of these works, and propose an approach that combines explicit keypoint encoding with standard image compression. The keypoint information is transmitted as side information. We perform experiments with JPEG, H.264/AVC Intra, and H.265/HEVC Intra, and present the results as plots which show the total bitrate versus the number of successfully matched features.

The remainder of this paper is organized as follows. In Section 2, the feature-preservation performance of JPEG, H.264/AVC Intra, and H.265/HEVC Intra as a function of bitrate is presented, and the motivation of our approach is explained in greater detail. Section 3 presents relevant statistics for keypoint encoding. In the following experiments, the encoding levels are set according to the statistics observed. In Section 4, several ideas to further reduce the number of keypoints to be encoded are explained. Section 5 presents experimental results showing that the proposed approach achieves a noticeable bitrate reduction compared with pure image compression at the same feature matching performance. Conclusions are presented in Section 6.

2. IMPACT OF IMAGE COMPRESSION ON FEATURE QUALITY

In this section, we investigate the impact of image compression on feature quality. We first extract features from images which are encoded with JPEG, H.264/AVC Intra, or H.265/HEVC Intra. These features are then matched to features which have been extracted from a set of uncompressed reference images. We plot the number of matches as a function of bitrate, and compare the results for the different standards.

2.1. Experimental settings

The Stanford mobile augmented reality (MAR) dataset [10] is used in our study. This dataset comprises 23 videos and 23 corresponding reference images. Each video has 100 frames, recorded with a resolution of 640 $\times$ 480 at 30 fps. Similar to [10], we use eight video sequences (*OpenCV*, *Wang Book*, *Barry White*, *Janet Jackson*, *Monsters Inc*, *Titanic*, *Glade*, and *Polish*), for pairwise feature matching evaluation. SIFT features are extracted and the Vlfeat [18] SIFT implementation is used. Similar to [10], the top 200 features are selected for each frame according to the CDVS Test Model.

For the evaluation of matched descriptors, the nearest neighbor distance ratio (NNDR) is used. For a query descriptor d extracted from a compressed test image, the nearest descriptor d_a and the second nearest descriptor d_b from the reference image are found, and thresholding on the distance ratio between them is applied. The query descriptor d and the nearest one d_a are matched if $||d - d_a||/||d - d_b|| < t$. We calculate the Euclidean Distance for descriptors and set t to 0.8 [6]. Afterwards, we use random sample consensus (RANSAC) [19] to remove incorrectly matched features assuming an affine transformation between the reference image and a test frame.

2.2. Feature matching performance for different image compression standards

The images from the eight test video sequences are encoded using JPEG with quality values 4, 8, 12, and 16 (Matlab, grayscale), the JM reference software (version 18.4) [20], and the HEVC Test Model (version 16.0) [21] with QP values 38, 42, 46, and 50. Then, SIFT features are extracted from the compressed images, and compared with the SIFT features extracted from the corresponding reference images. The number of matched features after applying RANSAC and the bitrates for the encoded images are averaged for all test images. The solid red, green, and blue curves in Fig. 1 show the feature matching performance for JPEG, H.264/AVC Intra, and H.265/HEVC Intra encoded images. Due to the superior rate-distortion quality of H.265/HEVC, the H.265/HEVC Intra encoded images lead to much better performance than the images encoded with JPEG and H.264/AVC Intra in terms of the number of matches at a given bitrate.

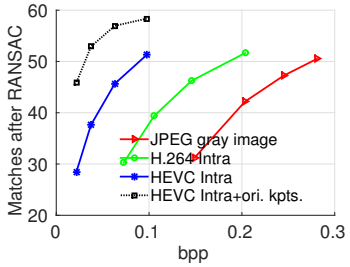

Fig. 1. SIFT feature matching performance for different standards. The dashed line is the result obtained for descriptors calculated on the H.265/HEVC Intra encoded images using the original keypoints from the uncompressed images.

2.3. Sensitivity of keypoints and descriptors

Given an input image I, the detected keypoints are expressed as.

$$k_i = \{l_i, \sigma_i, \theta_i\} \quad (1)$$

where l_i is the location of keypoint i in the image, σ_i is the scale, and θ_i is the orientation of the keypoint. The descrip-

tors obtained using the keypoints detected in image I are expressed as.

$$d_i = \Psi(k_i|I) \tag{2}$$

Similarly, the descriptors extracted from the compressed image $\overline{I}$ can be expressed as.

$$\overline{d_i} = \Psi(\overline{k_i}|\overline{I}) \tag{3}$$

Here, $\overline{k_i}$ are the keypoints detected in $\overline{I}$. We perform a simple experiment to show that the keypoint detection is easily affected by compression artifacts, and that in contrast, the descriptor is more robust. To demonstrate this, we calculate the descriptors from the compressed images using the keypoints extracted from the original (uncompressed) images. This can be expressed as.

$$d'_i = \Psi(k_i|\overline{I}) \tag{4}$$

This means that features d'_i have exactly the same keypoints k_i as in the uncompressed case. This allows us to ignore inaccurate keypoint detection from the compressed image and exclusively evaluate the descriptor robustness in the presence of compression artifacts. As the original keypoints k_i are idealistic and cannot be obtained from the compressed image $\overline{I}$, we show the obtained matching result for H.265/HEVC Intra encoding as a dashed line (upper bound) in Fig. 1. The results indicate that if the original keypoints are preserved, the descriptor vectors can be matched well despite strong compression. This observation motivates us to encode keypoints, and send them as side information with the compressed image.

3. KEYPOINT QUANTIZATION AND ENCODING

Since the location, scale, and orientation of keypoints are represented using floating-point numbers, we need to quantize them for efficient transmission. We apply different quantization methods for location, scale, and orientation, respectively. This section first presents experimentally determined statistics for keypoint quantization. From these results, we can determine for which quantization coarseness, we still get similar matching results as for unquantized keypoints.

3.1. Keypoint quantization

For locations, [15] uses a spatial grid over the original image with step sizes of 4, 6, and 8 pixels. Another work [16] quantizes locations into integer values, which means they are quantized on a grid with step size 1 pixel. The locations are hence quantized as follows.

$$\tilde{l_i} = f \cdot round(\frac{l_i}{f}) \tag{5}$$

where f is the quantization factor. In our experiments, we determine what quantization factor should be used such that the quantized locations do not significantly affect the number of feature matches. In our experiment, f varies from 1 to 5.

In SIFT, the scale σ_i can be represented as follows.

$$\sigma_i = \sigma_0 2^{(o+s/3)} + \Delta\sigma \tag{6}$$

where σ_0 is a base scale offset (i.e., 2.0159). o is an octave ranging from 0 to a number x depending on the size of the image (x is less than 4 for our dataset), and s is the integral scale in the range [0, 2]. Thus, 3 and 2 bits are sufficient to represent o and s, respectively. $\Delta\sigma$ is an offset calculated to increase the accuracy of the scale estimates in SIFT keypoint detection. This process fits a quadratic polynomial with the values of the detected scale space extremum ($\sigma_0 2^{(o+s/3)}$) to localize the more accurate scales with a resolution that is higher than the scale sampling density. Thus, the value $\Delta\sigma$ is related to the scale space extremum ($\sigma_0 2^{(o+s/3)}$). We calculate the difference between the detected scale σ_i and its corresponding scale space extremum ($\sigma_0 2^{(o+s/3)}$). We then normalize the difference as follows.

$$\Delta\sigma_n = (\sigma_i - \sigma_0 2^{(o+s/3)})/\sigma_0 2^{(o+s/3)} \tag{7}$$

Fig. 2 shows the magnitudes and the distribution of $\Delta\sigma_n$ for the features extracted from all eight video sequences. From the magnitudes (left plot) we can see that the values of the normalized differences are mainly within [-0.2, 0.2]. The distribution on the right in Fig. 2 has approximately a Gaussian form. We encode the normalized difference using the Lloyd-Max quantization algorithm. The codebook and partition are trained from these statistics. We assign 3, 2, 1, and 0 bits for $\Delta\sigma_n$, respectively, where 0 bit means we use the $\sigma_0 2^{(o+s/3)}$ without adding any offset. Thus, the scales including o, s and $\Delta\sigma_n$ are assigned 8, 7, 6, and 5 bits in total for testing.

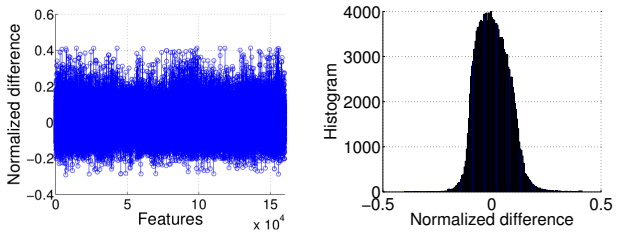

Fig. 2. Magnitude and distribution of $\Delta\sigma_n$ for 160000 features extracted from eight test video sequences.

Similar to the location quantization, the orientations are quantized as follows.

$$E(\theta_i) = round((\frac{\theta_i}{2\pi} + 0.75) \times (2^t - 1))$$

$$\tilde{\theta_i} = \Big(\frac{E(\theta_i)}{2^t - 1} - 0.75\Big) \cdot 2\pi \tag{8}$$

where 0.75 is an offset that ensures that the values $(\frac{\theta_i}{2\pi} + 0.75)$ are within [0, 1) in the Vlfeat SIFT implementation. Then, the

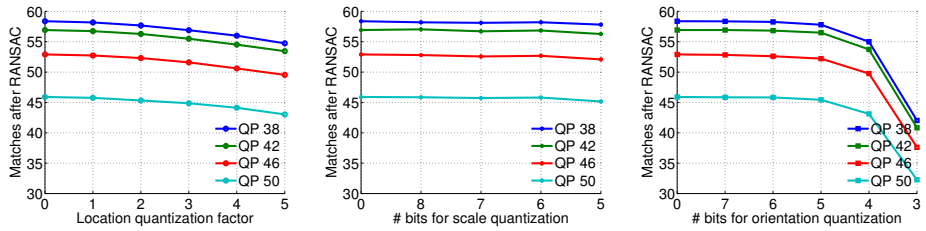

Fig. 3. The impact of keypoint quantization. The value 0 on the horizontal axis indicates the idealistic uncompressed keypoints.

index $E(\theta_i)$ can be represented by values within [0, 2^t) and can be encoded by fixed length coding with t bits. In our test, t is set to 7, 6, 5, 4, and 3 which quantize the orientations into 128, 64, 32, 16, and 8 levels, respectively.

Fig. 3 shows the impact of keypoint quantization on the matching performance. In order to be able to individually evaluate the impact of quantization on one quantity (e.g., the location), the other quantities (e.g., the scale and orientation) use the original float-pointing values. The matching performance for uncompressed keypoints is shown at the very left of each plot (value 0 on the horizontal axis). From the left plot it can be seen that the matching performance starts dropping at a quantization factor of 2 for locations. In the middle plot, we can see that the scale quantization does not affect the matching performance even when using only 6 bits. In the right plot, the performance drops when encoding with 5 bits for orientations.

Our goal is to select reasonable quantization levels such that the matching performance is not affected and the number of bits required should also be small. Thus, we choose the location quantization factor 1 (see left plot), 6 bits for the scale (see middle plot), and 6 bits for the orientation (see right plot) in the following experiments. As a result, the quantized keypoints lead to feature matching performance that is similar to the idealistic uncompressed keypoints. In total, the encoding of scale and orientation for one keypoint requires 12 bits.

3.2. Context model for location coding

As explained in the previous section, we quantize the original locations into integer values (quantization factor 1). To encode quantized locations, we use the same approach used in CDVS [7], i.e., sum-based context-based arithmetic coding. The number of bits for location encoding depends on the distribution and number of keypoints to be encoded. In sum-based context-based arithmetic coding, we must first train the context model. Similar to [15] and CDVS [7], we use the Inria Holidays Dataset[1] and the Caltech Building Dataset[2] for training. Note that the joint dataset comprises 1741 images. Since the location quantization factor is set to 1, the block width is correspondingly set to 1. Unlike the fixed number of features to be encoded in the CDVS Test Model, we will encode a different number of keypoints (K^3, see Table 1) for each image in the following experiments. Thus, we test two sets to determine the context range (i.e., 200 features and 100 features) from each image, and encode these locations. For simplicity, we use the same test dataset, and encode the quantized locations again after obtaining the context model. The context range is selected from 1 to 60, and the average number of bits used for encoding the locations are recorded. For both sets, the curves flatten beyond a context range of 45 with a minimum at 49. Thus, we select a context range of 49, and the quantized locations are encoded using the correspondingly trained model.

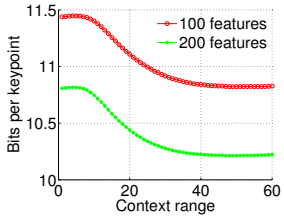

Fig. 4. Average number of bits required for keypoint location encoding as a function of context range.

3.3. Keypoint set

We will encode keypoints using the previously observed quantization levels for locations, scales, and orientations. Here, we express the set of all original keypoints as $K = \{k_1, k_2, ..., k_N\}$ and the set of all original descriptors as $D = \{d_1, d_2, ..., d_N\}$, where N is the number of keypoints (i.e., 200). Similarly, we express the set of all quantized keypoints as $K^1 = \{k_1^1, k_2^1, ..., k_N^1\}$ and the extracted descriptor set as $D^1 = \{d_1^1, d_2^1, ..., d_N^1\}$. In the next section, we show that some keypoints in the set K^1 can be removed before transmitting them to the server, which leads to a further rate reduction for the side information.

4. KEYPOINT SELECTION AND TRANSMISSION

We propose two steps to reduce the number of keypoints to be encoded in order to further lower the bitrate required for the side information.

We first remove spurious keypoints in K^1. The definition of spurious keypoints is that the descriptors extracted

[1]http://lear.inrialpes.fr/people/jegou/data.php
[2]http://vision.caltech.edu/malaa/datasets/caltech-buildings/

around those keypoints from the compressed image $\overline{I}$ cannot be matched to the descriptors extracted from the original (uncompressed) image I. The remaining keypoints are expressed as follows.

$$K^2 = \left\{k_i^1 \in K^1 | M(d_i^1, d_i) = 1\right\} \tag{9}$$

where $M(\cdot,\cdot) = 1$ denotes the matched pairs of d_i^1 and d_i (see (2)). Please note that the matching process checks two aspects for a pair of features from d_i^1 and d_i. First, the descriptors should be matched according to the NNDR strategy. Second, the locations of the pair should be within 1.5 pixels in our experiments. If the pair meets the two requirements, they are considered as a matching pair.

In the previous step, we only consider the keypoints derived from the original image I. Applying the detector algorithm to the compressed image $\overline{I}$ also generates some keypoints. Some of them are good enough to produce correct descriptors relative to the descriptors from the uncompressed image, especially when the compressed image $\overline{I}$ is of good quality. Thus, we propose to remove duplicated keypoints in K^2 relative to the keypoints $\overline{K}$ (see (3)) extracted from the compressed image $\overline{I}$ itself. We express the useful keypoints K^c from the compressed image as follows.

$$K^c = \{\overline{k_i} \in \overline{K} | M(\overline{d_i}, d_i) = 1\} \tag{10}$$

Thus, the remaining keypoints are expressed as.

$$K^3 = \{K^2 \setminus K^c\},\ thus,\ K^3 \subseteq K^2 \tag{11}$$

The information C about whether a keypoint from the compressed image is a useful one is encoded by binary arithmetic coding, and the resulting bitrate is added to the total keypoints rate. Note that after removing the duplicated keypoints, the remaining keypoints K^3 and C are encoded and transmitted as side information.

5. KEYPOINTS AT THE SERVER

At the server side, the keypoints K^3 are decoded, and the useful keypoints K^c are selected from the keypoints extracted from the compressed image itself using the information C. These two sets of keypoints are then combined, and the descriptors are calculated around these keypoints.

$$K^* = \{K^3 \cup K^c\},\ D^* = \Psi(K^* | \overline{I}) \tag{12}$$

where K^* is the final set of keypoints for which the descriptor set D^* is extracted from the compressed image $\overline{I}$.

6. EXPERIMENTAL RESULTS

Table 1 shows the average number of keypoints in the sets K^1, K^2, and K^3. As described above, K^2 are the keypoints after removing spurious ones relative to K^1. When the QP value is large, the set K^2 is small due to strong compression artifacts that lead to many spurious features. K^3 are the keypoints after dropping duplicated ones relative to the keypoints obtained from the compressed image itself. When the QP value is small, the detector extracts many useful keypoints from the compressed image itself. Thus, the remaining keypoints K^3 to be encoded are relatively few. As illustrated in (10) and (11), at the server side, the useful keypoints K^c can be obtained from the keypoints $\overline{K}$ from the compressed image itself and the information C. The keypoints K^3 and K^c are combined, and the number of keypoints K^* is identical to the number of keypoints in set K^2.

Table 1. Average number of keypoints per frame

QP	38	42	46	50
K^1	200	200	200	200
K^2	192.6	189.0	178.8	155.2
K^3	90.0	111.9	130.1	131.4
K^*	192.6	189.0	178.8	155.2

Fig. 5 shows the original keypoints K in red and the combined keypoints K^* in green. Most of them overlap, indicating that our proposed keypoints are almost the same as the original ones. However, a few original keypoints have no counterparts. Thus, the descriptors calculated from these keypoints are spurious and discarded to save bits.

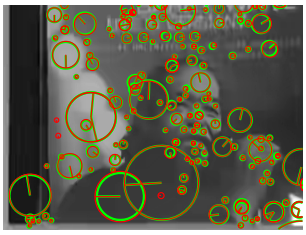

Fig. 5. The original keypoints K are in red and the combined keypoints K^* at the server side are shown in green.

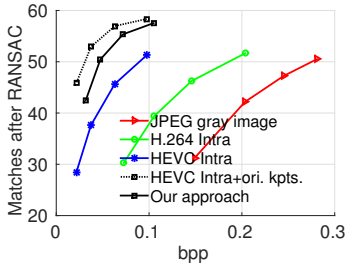

Fig. 6. Matching performance comparison for different approaches. The proposed approach is shown using the solid black line. The bitrate includes the encoded keypoints and the H.265/HEVC Intra encoded image.

The solid black line in Fig. 6 shows the final matching performance of our proposed approach as a function of the total bitrate used for the compressed image plus the side information. We can see that the number of matches is significantly

improved. At an average of approximately 42 matches per image, the proposed approach achieves a 6.5 $\times$ bitrate reduction compared with JPEG, a 3.9 $\times$ bitrate reduction compared with H.264/AVC Intra, and a 1.6 $\times$ bitrate reduction compared with H.265/HEVC Intra. Note that there is a trade-off between the matching performance and the quantization levels for locations, scales, and orientations. We only present the results obtained using the selected quantization levels, which seem to be reasonable for improving the matching performance.

7. CONCLUSION

While image compression artifacts have an adverse effect on feature matching performance, the situation can be significantly improved by using the original SIFT keypoints. We propose a novel approach for feature-preserving image compression, where the encoded keypoints are signaled as side information with the compressed image. Our results show that the proposed approach significantly improves the matching performance. Besides the merit that the decoded images can be watched or stored, another advantage of our feature-preserving image compression approach is that the orientations and scales can be used for geometric verification. In our future work, we plan to apply the idea to videos using keypoint prediction and differential keypoint coding.

8. REFERENCES

[1] V. Chandrasekhar, M. Makar, G. Takacs, D. Chen, S. S. Tsai, N. Cheung, R. Grzeszczuk, Y. Reznik, and B. Girod, "Survey of SIFT Compression Schemes," in *Proc. Second International Workshop on Mobile Multimedia Processing (WMMP)*, Istanbul, Turkey, Aug. 2010.

[2] L. Baroffio, M. Cesana, A Redondi, M. Tagliasacchi, and S. Tubaro, "Coding Visual Features Extracted From Video Sequences," *IEEE Transactions on Image Processing*, vol. 23, no. 5, pp. 2262–2276, May 2014.

[3] M. Makar, V. Chandrasekhar, S. S. Tsai, D. Chen, and B. Girod, "Interframe Coding of Feature Descriptors for Mobile Augmented Reality," *IEEE Transactions on Image Processing*, vol. 23, no. 8, pp. 3352–3367, Aug. 2014.

[4] A Redondi, L. Baroffio, M. Cesana, and M. Tagliasacchi, "Compress-then-analyze vs. analyze-then-compress: Two paradigms for image analysis in visual sensor networks," in *IEEE International Workshop on Multimedia Signal Processing (MMSP)*, Sept. 2013, pp. 278–282.

[5] A Redondi, L. Baroffio, J. Ascenso, M. Cesana, and M. Tagliasacchi, "Rate-accuracy optimization of binary descriptors," in *IEEE International Conference on Image Processing*, Sept. 2013, pp. 2910–2914.

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

[7] *http://mpeg.chiariglione.org/standards/mpeg-7/compact-descriptors-visual-search*.

[8] M. Makar, C.-L. Chang, D. Chen, S. Tsai, and B. Girod, "Compression of image patches for local feature extraction," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, Taipei, Taiwan, Apr. 2009, pp. 821–824.

[9] M. Makar, H. Lakshman, V. Chandrasekhar, and B. Girod, "Gradient preserving quantization," in *Proc. IEEE Int. Conf. Image Processing*, Orlando, Florida, USA, Sept.-Oct. 2012.

[10] M. Makar, S. S. Tsai, V. Chandrasekhar, D. Chen, and B. Girod, "Interframe Coding of Canonical patches for Low Bit-rate Mobile Augmented Reality.," *Int. J. Semantic Computing*, vol. 7, no. 1, pp. 5–24, 2013.

[11] L. Duan, X. Liu, J. Chen, T. Huang, and W. Gao, "Optimizing JPEG quantization table for low bit rate mobile visual search," in *Proc. Visual Communications and Image Processing*, San Diego, CA, USA, Nov. 2012, pp. 1–6.

[12] J. Chao, H. Chen, and E. Steinbach, "On the design of a novel JPEG quantization table for improved feature detection performance," in *IEEE International Conference on Image Processing*, Melbourne, Australia, Sept. 2013.

[13] J. Chao and E. Steinbach, "Preserving SIFT features in JPEG-encoded images," in *Proc. IEEE Int. Conf. Image Processing*, Brussels, Belgium, Sept. 2011, pp. 301–304.

[14] J. Chao, R. Huitl, E. Steinbach, and D. Schroeder, "A Novel Rate Control Framework for SIFT/SURF Feature Preservation in H.264/AVC Video Compression," *IEEE Transactions on Circuits and Systems for Video Technology (accepted for publication)*, 2014.

[15] Sam S. Tsai, D. Chen, G. Takacs, V. Chandrasekhar, M. Makar, R. Grzeszczuk, and B. Girod, "Improved coding for image feature location information," in *Proc. SPIE*, 2012, vol. 8499.

[16] H. Yue, X. Sun, J. Yang, and F. Wu, "Cloud-Based Image Coding for Mobile Devices - Toward Thousands to One Compression.," *IEEE Transactions on Multimedia*, vol. 15, no. 4, pp. 845–857, 2013.

[17] H. Yue, X. Sun, F. Wu, and J. Yang, "SIFT-Based Image Compression," in *IEEE International Conference on Multimedia and Expo*, 2012, pp. 473–478.

[18] *http://www.vlfeat.org/*.

[19] M. Fischler and R.C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[20] *http://iphome.hhi.de/suehring/tml/*.

[21] *http://hevc.hhi.fraunhofer.de/*.