Low Bit-Rate Compression of Facial Images

Michael Elad   Roman Goldenberg   Ron Kimmel
Computer Science Department
Technion – Israel Institute of Technology
Haifa, 32000, Israel
E-mail: elad/romang/ron@cs.technion.ac.il

Abstract

An efficient approach for face compression is introduced. Restricting a family of images to frontal facial mug-shots enables us to first geometrically deform a given face into a canonical form in which the same facial features are mapped to the same spatial locations. Next, we break the image into tiles and model each image tile in a compact manner. Modeling the tile content relies on clustering the same tile location at many training images. A tree of vector-quantization dictionaries is constructed per location, and lossy compression is achieved using bit-allocation according to the significance of a tile. Repeating this modeling/coding scheme over several scales, the resulting multi-scale algorithm is demonstrated to compress facial images at very low bit rates while keeping high visual qualities, outperforming JPEG-2000 performance significantly.

Keywords

Image compression, facial images, geometric canonization, vector quantization.

I. Introduction

The problem of image compression has been thoroughly explored for years and efficient general purpose compression algorithms are available today. Much less attention has been given to the problem of image compression for the case in which a strong prior is available for the class of images to be compressed. This happens, for example, when the input belongs to a certain, a-priori known and possibly very specific class of images. One expects that for such specific cases an even more efficient compression should exist, outperforming general purpose algorithms.

In this paper we address the problem of compressing human frontal facial images. The images we deal with are passport-type photos - full face, frontal view, plain background, no dark glasses, without hats and other non-standard clothing, see for example Figure 1.

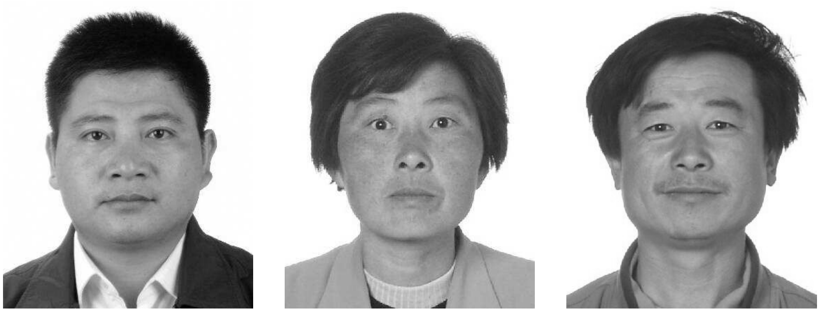


Fig. 1. Input examples - passport-type photos.

Our goal is a compression method of a standard digital 358×441 b/w passport photograph (154KBytes at 8 bits per pixel) into less than 1KByte representation (i.e. compression ratio of about $1 : 154$ and beyond). The goal of this note is to introduce a method to compress and decompress the facial image, so that it is visually appealing, at a quality sufficient to un-mistakenly visually identify a given subject.

The approach we take is based on the following concepts.

- **Geometrical Canonization:** Restricted to frontal facial mug-shots, the handled images are geometrically deformed into a canonical form, in which facial features are located at the same spatial locations. Using a plain feature detection procedure, the image is divided to disjoint and covering set of triangles, each deformed using a different affine warp.
- **Clustering:** The overall treatment of the images is local, by splitting the image into tiles. Every tile is coded using vector quantization. A flexible bit allocation is permitted, by using tree VQ [1], [2], [3].
- **Hierarchial Multi-Scale Treatment:** The coding is performed on a pyramidal representation of the image, processing the information from coarse to fine, and at each stage operating on the image residual.

The proposed compression algorithm is demonstrated and compared to JPEG-2000 [20]. Compressed facial images at very low bit rates are shown to keep high visual qualities (compression ratio of $154 : 1$ with an average PSNR of $30dB$), outperforming JPEG-2000 results.

In Section II we review existing contributions in the literature referring to compression of facial images. Section III presents a detailed description of the geometric canonization, which turns out to be crucial for the compression performance. In section IV we discuss the core coding part of the algorithm that is based on VQ, covering the training and the testing phases, along with the extension to a multi-scale scheme. Experiments and results are shown in Section V, and conclusions are drawn in Section VI.

II. Related work and the proposed approach

Face images are common, and as such are extensively studied in the literature especially in the context of detection and recognition. Remarkably, among the many thousands of papers that discuss ways to compress still images in general, only few address the compression of face images [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19]. Assumptions on the images' content, and tailoring compression algorithm that exploit these assumptions, lead these contributions to a variety of solutions. In this section we briefly describe such methods and their rationales.

The importance of geometric pre-analysis of the image to the compression performance has been recognized in several papers [5], [6], [7], [8], [9], [10], [11], [15], [17]. Most of these methods use feature detection for locating semantic landmarks, such as eyes, nose, mouth, etc. Once found, one can either warp the image to a canonical configuration, as done in [5], [7], [8], [9], [11], or operate on the original image, while adapting the treatment spatially based on the content detected [6], [10], [15].

Here, we first deform the image into a canonical form. Alignment in most papers has been implemented using a rigid transformation (rotation, scale, shift) [5], [8], [9], [11], [15], thus limiting the accuracy of the fit. In our method we employ a more delicate canonization that leads to a perfect alignment of 13 points[1], thereby boosting compression performance, as described in the next section.

Coding of the image content can be done in various ways. An early paper by Moghaddam and Pentland [5] applied a global Principal Component Analysis (PCA), training a transform that leads to optimal compactization of the image energy in the leading transform coefficients. Truncating and quantizing these coefficients leads to the desired coding. PCA (or Karhunen Loeve Transform (KLT)) has been also practiced in [11]. However, rather than considering global basis functions that span over the entire image support, the PCA is done on small tiles of size 8×8 pixels. For every image, all tiles are clustered into 4 groups based on their local activity, and per each, a PCA is trained. As this training varies from one image to another, the obtained transform matrices must be sent as side information.

The papers by Ferreira and Figueiredo [13], [14], [19] also describe a transform training and processing of small tiles. They propose the use of Independent Component Analysis (ICA) for the representation of image tiles. However, their adaptation to the image content is less strict, as the learned transform is assumed to be the same for all tiles, regardless of their spatial location. In this context, their view matches the JPEG approach that employs the same transform to all the blocks in the image. As such, their scheme is more general than the one we consider, and geometrical alignment is irrelevant. While the compression performance reported in these papers is better than JPEG, and comparable to JPEG-2000, it is clear that such semi-adaptation is destined to be inferior to methods that also adapt the transform spatially.

Getting closer to the proposed approach, the reported compression algorithm in [8], [9] consists of an encoding stage that is based on the wavelet transform, followed by vector

[1] This number of features was chosen as a good compromise between three criteria: (i) the desire to use reliable and detectable features; (ii) the desire to use as many as possible points so as to align the images better; and (iii) the desire to use only few features to reduce their side-information cost.

quantization of different bands. VQ is also used in [15], where it is directly applied on several image features, instead of arbitrary slicing the image into tiles. As mentioned earlier, while our coding is also done using VQ, the adaptation we propose is more flexible, as every tile location is trained separately. The various obtained dictionaries are stored in the encoder and the decoder, and thus they are not needed as side information. More information on this scheme is given in Section IV.

A somewhat different representation that exploits both spatial and inter-image dependencies is considered in [12], [16], [18]. Treating the group of images as a 3D tensor, its decomposition to three-way rank-one approximation is proposed. Generalizing the Singular Value Decomposition (SVD) in several possible ways, these attempts are claimed to lead to more efficient compression. Since these papers consider only global decomposition, their results are inferior to a tile-based treatment, as studied here.

Performance-wise, most of the above algorithms are shown to outperform the JPEG and become similar to, or just slightly better than the JPEG-2000 standard. Among these papers, those published before the year 2000 do not compare to JPEG-2000. It is hard to give conclusive comparison to these methods as most of them consider small images (less than 100 × 100 pixels) and relatively high rate (above 0.1bpp), while we consider larger images and a much lower rate (0.05bpp).

III. Alignment

As our solution exploits the similarity between corresponding regions in facial images, the images need to be aligned first. The input image is geometrically transformed into a canonical form, which brings it as close as possible to a predefined "average" facial image. For this goal we apply a feature-based correspondence to map the input image to its canonical form.

Before turning to describe the alignment procedure, we note that while it is tempting to exploit the symmetry of faces, extensive study that we performed shows that such symmetry cannot be trusted, and it often leads to inferior results. Apparently, mirroring ones' half-face generates high frequency de-correlated errors due to the subtle asymmetry of faces (and the fact that these images are not exactly frontal ones). Thus, edges fall near (and not on-top of) edges, and the result would be a waist of significant bits required to transmit large prediction error.

A. Facial features detection

We define a set of thirteen facial features. Six of them are anchored to facial anatomical landmarks - eyes, nose, mouth and chin (see Figure 2), and the rest are along the face outline.

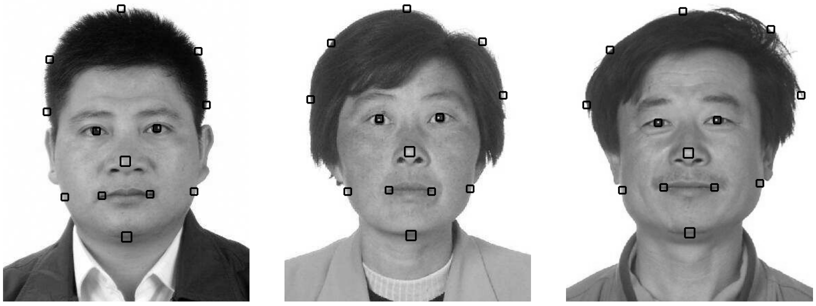


Fig. 2. Facial feature points. There are two features in the eyes that cannot be seen in these images.

The outline of the face is detected by background subtraction. For the plane background, as in our case, we take samples of the background color in several image locations and construct a linear background model for the whole image to compensate for non-uniform illumination. The foreground is then detected by thresholding, followed by morphological filtering.

One issue that requires special attention is the hair around the face, which can significantly alter the form of the facial outline. In order to cope with this problem we estimate the facial skin tone color and correct the outline contour location around the face inwards, thereby avoiding the hair region. In Figure 3 one can see how the hairline affects the location of facial outline markers at the mouth level (Top). The markers are brought back to the face line by the hair correction procedure (Bottom).

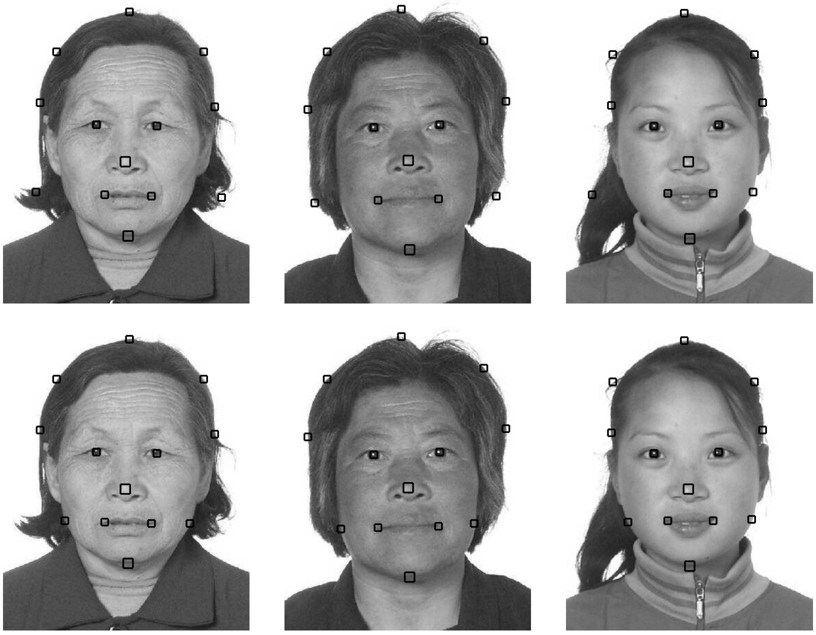


Fig. 3. Facial features detection with (Bottom) and without hair correction (Top).

The internal facial features are detected using correlation based matching. The correlation kernels are built by averaging relatively small image window around the feature point over the training set. See Figure 4.

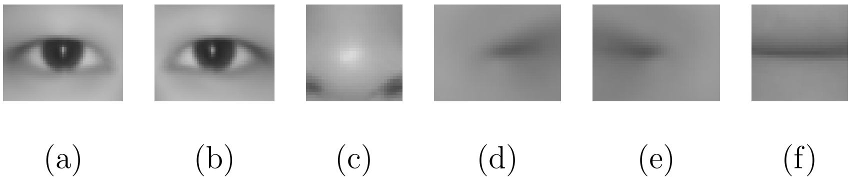


Fig. 4. Correlation masks for facial features detection - (a) left eye, (b) right eye, (c) nose tip, (d) left mouth corner, (e) right mouth corner, (f) chin.

B. Image warping

At the training phase we scan a large set of images, detect facial features as described above, and find the average feature locations. Given a set of corresponding feature points in the input image and the reference "average" facial image, we need to warp the former onto the latter. This is done using a piecewise affine transform.

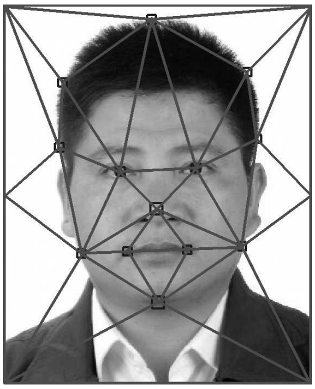


Fig. 5. Affine warping triangulation.

The set of thirteen feature points together with six points at the image corners and boundaries define a triangulation in the image domain (see Figure 5). Every triangle $\triangle A_iB_iC_i$ in the input image and the corresponding triangle $\triangle ABC$ in the "average" image uniquely define an affine transform T such that

$$T[A_iB_iC_i]^T = [ABC]^T. \tag{1}$$

Then, for every $x \in \triangle A_iB_iC_i$

$$I_{warped}(x) = I(Tx), \tag{2}$$

where I and I_{warped} are the input and the aligned images respectively. Figure 6 shows several examples of aligned images.

C. Implementation considerations

The proposed feature detection process as described in Section III.A may fail to find the proper locations, thus jeopardizing the overall coding process. In our experiments we found that more than 99% of the images were treated properly by the automatic detection

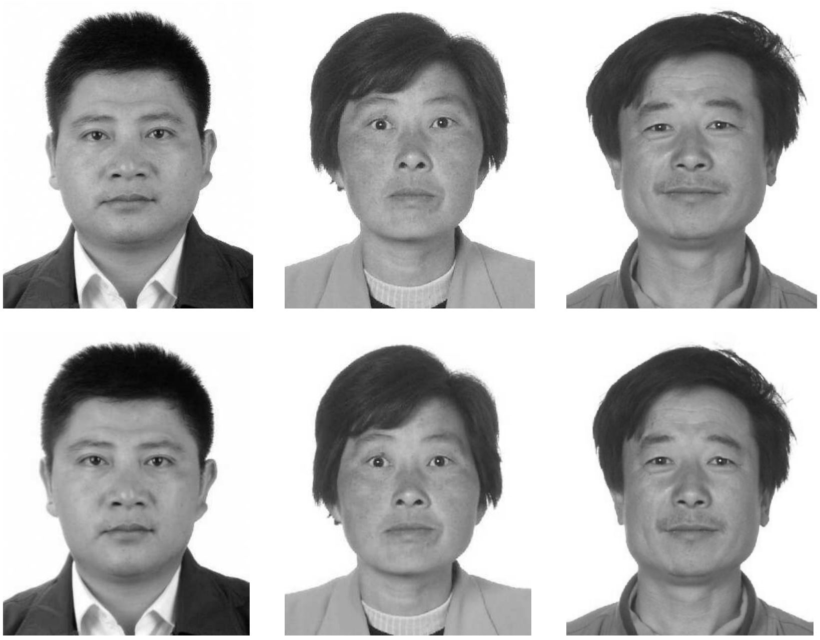


Fig. 6. Input images (Top) and their aligned (canonical) versions (Bottom)

procedure. Identification of the errors for the training images ($6,000$ images altogether) was done manually, and their features were marked semi-manually (getting a proposed location from the system and updating it if needed).

In the coding of new test images, failure to detect the features leads to extreme low PSNR, and thus it is easily detected automatically. For such images, features are required to be marked by the user. Alternatively, such images can be coded using regular JPEG-2000 (or any other competing scheme).

As to the geometrical canonization as described in Section III.B, it is done on all the training images, before proceeding to the coding stage. This canonization is also done when encoding a test image. Since the decoder should apply the inverse transform, side information containing the 13 feature coordinates is required. This requires less than 20Bytes, using a prior knowledge on the coordinates' distribution.

IV. Coding strategy

A. Choice of VQ

Once the images are aligned, one would expect a certain similarity between the corresponding regions, allowing for compact representation. Here, we can explore one of several principal approaches to exploit this similarity for compression. The use of PCA, as described in Section II, can lead to such modeling and coding. However, as we limit our discussion to extremely low bit-rates, keeping a number of reasonably quantized coefficients even for a relatively small basis may become prohibitively expensive. Therefore, we implemented local clustering and vector quantization (VQ), which fits well our low-bit-rates [1], [2], [3].

B. Training the coder

The images are divided into small blocks of size 8×8 (a parameter of the algorithm). For each block location we have $6,000$ examples taken from training images. The K-Means algorithm is applied to find the best 2^k representation vectors, where k is the number of bits allocated for this block. The value of k is chosen as the minimal number of bits yielding a mean-square-error (MSE) over the training set lower than a predefined threshold. Actual values of k in our tests are in the range $[0, 10]$.

In order to save even more bits, the MSE thresholds for each block are chosen to provide lower error in recognition-critical portrait areas (eyes, nose, etc.), at the expense of discriminating less visually important regions (background, clothing, etc.). This idea has also appeared in [17], where the coding was done using JPEG-2000, but in a spatial selective way. Figure 7(a) shows the root mean-squared-error (RMSE) threshold allocation map used for training, with the requested maximal allowed error varying from 6 to 15 intensity levels. The actual bit allocation per block established by the training process does not necessarily correlate with the RMSE threshold map, as seen in Figure 7(b). This is because the system is able to represent regions with low variability, e.g. forehead or cheeks, with a small amount of bits even when the allowed error for this region is low.

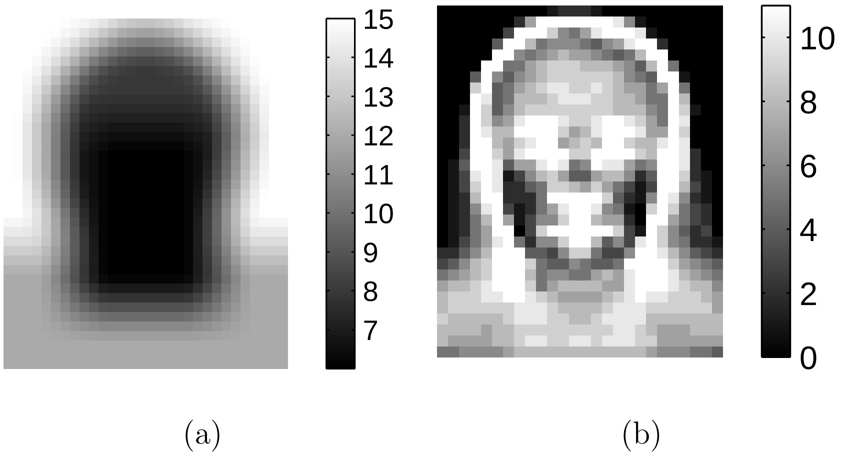


Fig. 7. (a) RMSE threshold map (intensity levels), (b) Bit allocation map (bits per block).

We use 8 × 8 blocks with an overlap of one pixel to minimize blockiness effects. When reconstructed, the pixels at the overlapping regions are taken as average between the overlapping blocks. An alternative approach that removes this overlapping and applies deblocking as a post-processing in the decoder is possible, but was not pursued.

Handling of color RGB images is simple. Such images are first converted into $YCbCr$ format and the training is performed independently for each one of the three channels, while the chroma components are taken at half of the original resolution along each axis, and the MSE thresholds are chosen to be less restrictive. The computed quantization vectors are then stored for each block and each channel and used both for encoding and decoding.

C. Encoding and decoding

The encoding process starts with alignment - facial features are detected as described above and the image is warped to its "canonical" form. Then, each tile is quantized using VQ, finding the closest representation vector (in the MSE sense) among the VQ vectors stored for this tile. The chosen VQ vector's index serves as a representation for the block.

The process should be repeated for all 3 image channels in case of color images.

The bit stream that represents the compressed image contains data on facial features coordinates and the VQ vectors indices. We did not implement an entropy coding stage for further compression. There are several reasons for this choice (i) we found experimentally that relatively simple entropy coding schemes (Huffman, LZ) hardly provide any further compression; (ii) we wanted to keep the algorithm simple and avoid complex entropy coding schemes that might provide further compression; and (iii) the performance even without such schemes was found to be sufficient.

For decoding, the steps described above are performed in a reversed order, namely, the bit stream is parsed into the feature points coordinates and the VQ vector indices. As the bit allocation and the order of fields in the bit stream is known and fixed, no additional formatting symbols are required for parsing.

For a given tile, the image is retrieved by the index from the VQ vectors set stored for this location. Overlapping block pixels are averaged, and the process is repeated for all three image channels. Finally, the image is warped using an inverse alignment stage. The locations of the feature points in the input image retrieved from the bit stream along with the known feature points coordinates in the canonical image, uniquely define the inverse piecewise affine transform to be applied to the restored image.

D. Multi-scale approach

A multi-scale approach can easily be incorporated into the proposed scheme. The idea is to use larger size tiles/blocks for correlated image regions. That is, instead of coding several neighboring correlated small blocks separately, one can apply the VQ analysis for the whole correlated area and use only one vector index to represent it.

This gain can be practically achieved by operating using a constant block size (8×8 as described above) over all the layers of a Gaussian pyramid of the input image [4]. Such a pyramid leads to a set of s resolution layers, denoted as I_{s-1}, I_{s-2}, $\ldots$, I_0. The image I_{s-1} is the coarsest layer, smaller by factor 2^{s-1} in each axis compared to the original, and containing a reduced (smoothed and down-sampled) version of the original. The rest of the layers are similar, growing bigger by a factor of 2 along each axis, and I_0 is the finest layer, being the original image.

The proposed multi-scale framework starts with I_{s-1}, applying the training and the coding as described above, on patches of size 8×8 pixels. The decompressed image is interpolated, and subtracted from I_{s-2}. The residual is passed through the same training/coding stages. This repeats until the finest resolution layer is reached.

In coding and decoding across scales, care must be given to the bits-allocation in each layer. We use the MSE threshold map as described above, choosing K per each tile and each resolution layer to conform with the target error.

V. Experiments

In our experiments we used a two level multi-scale approach. The system was trained on $6,000$ images. By tuning the MSE thresholds we control the rate – i.e., vary the required number of bytes to represent the encoded image.

The VQ dictionaries trained should to be stored at the encoder and the decoder. For the rates tested in the following experiments we found that ≈ 40 MBytes are required. We note that this size can be reduced dramatically by introducing quantization of the dictionary entries. Our coding scheme was simulated using non-optimized Matlab software, and ran on a PC (Pentium 2, 1.5GHz, 1GByte RAM). Encoding of an image takes 2.7 seconds, and its decoding requires 0.8 seconds.

Figure 8 shows several examples of image compression using our method. This image was taken from the test set. For comparison, the results are presented along with the images compressed using the standard JPEG-2000 images compression algorithm. As claimed, the results using our method show better visual quality.

We conducted an experiment where 10 unprepared subjects where presented with 20 images compressed using our method and JPEG-2000 using $1 - 3$KBytes images. The respondents were asked to grade the proposed compression results relative to the JPEG-2000. The average grade indicated that our compression result with 1KByte falls somewhere between JPEG-2000 2kBytes and 3kBytes in terms of subjective visual appeal. Figure 9 presents the JPEG-2000 images obtained with 1 KByte, 2 KBytes, and 3 KBytes for the example shown in Figure 8, to illustrate this comparison.

In order to get a more objective measure of performance, we coded 1000 test images in varying rates, using both JPEG-2000 and our algorithm. Figure 10 shows the Rate-Distortion curves obtained with both methods. As can be clearly seen, while the developed algorithm generally shows better compression performance for the range of rates explored, its gain is pronounced as the rate decreases.

VI. Conclusions

A frontal facial compression method was presented and its advantages were explored. It was shown that a geometric warping into a canonical form, followed by an efficient coding for each block, allows compression rates that are almost twice better compared to the best commercial compression method namely JPEG-2000. This is not the end of the road, and VQ may well be found to be inferior to alternative ways of representing the block patches. In a sequel paper, we intend to explore the role of sparse representation in tailored dictionaries in representing image tiles.

Acknowledgement

The authors would like to thank Eyal Gordon for proposing the triangulation-based warping scheme implemented in this project.

References

[1] Y. Linde, A. Buzo, and R.M. Gray, An algorithm for vector quantiser design,*IEEE Transactions on Communications*, (28):84–95, January 1980.

[2] A. Gersho and R.M. Gray, *Vector Quantization And Signal Compression*, Kluwer Academic Publishers, Dordrecht, Netherlands, 1992.

[3] P. Cosman, R. Gray, and M. Vetterli, Vector quantization of images subbands: A survey, *IEEE Trans. Image Processing*, 5(2):202–225, 1996.

[4] P.J. Burt and E.H. Adelson, The Laplacian pyramid as a compact image code, *IEEE Transactions on Communications*, 31(4):532–540, 1983.

[5] B. Moghaddam and A. Pentland, An automatic system for model-based coding of faces, Proceedings of *DCC '95 Data Compression Conference*, Snowbird, UT, USA, 28-30 March 1995.

[6] J.H. Hu, R.S. Wang, and Y. Wang, Compression of personal identification pictures using vector quantization with facial feature correction, *Optical-Engineering*, 35(1):198–203, 1996.

[7] A. Lanitis, C.J. Taylor, and T.F. Cootes, Automatic interpretation and coding of face images using flexible models, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):743–756, 1997.

[8] M. Sakalli, H. Yan, K.M. Lam, and T. Kondo, Model-based multi-stage compression of human face images, Proceedings of 14th International Conference on Pattern Recognition (ICPR), Brisbane, Qld., Australia. 16-20 Aug. 1998.

[9] M. Sakalli and H. Yan, Feature-based compression of human face images *Optical-Engineering.* 37(5):1520-1529, 1998.

[10] J. Huang and Y. Wang, Compression of color facial images using feature correction two-stage vector quantization, *IEEE Trans. on Image Processing*, 8(1):102–109, 1999.

[11] M. Sakalli, H. Yan, and A. Fu, A region-based scheme using RKLT and predictive classified vector quantization, *Computer Vision and Image Understanding*, 75(3):269–280, 1999.

[12] A. Shashua and A. Levin, Linear image coding for regression and classification using the tensor-rank principle, Proceedings of *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Kauai, HI, USA, 8-14 Dec. 2001.

[13] A.J. Ferreira and M.A.T. Figueiredo, Class-adapted image compression using independent component analysis, Proceedings of *International Conference on Image Processing (ICIP)*, Barcelona, Spain, 14-17 Sept. 2003.

[14] A.J. Ferreira and M.A.T. Figueiredo, Image compression using orthogonalized independent components bases,

Proceedings of *IEEE XIII Workshop on Neural Networks for Signal Processing*, Toulouse, France, 17-19 Sept. 2003.

[15] O.N. Gerek and C. Hatice, Segmentation based coding of human face images for retrieval, *Signal-Processing*, 84(6):1041–1047, 2004.

[16] K. Inoue and K. Urahama, DSVD: a tensor-based image compression and recognition method, Proceedinsg of *IEEE International Symposium on Circuits and Systems (ISCAS)*, Kobe, Japan, 23-26 May 2005.

[17] Z. Qiuyu and W. Suozhong, Color personal ID photo compression based on object segmentation, Proceedings of *Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, Victoria, BC, Canada, 24-26 Aug. 2005.

[18] T. Hazan, S. Polak, and A. Shashua, Sparse image coding using a 3D non-negative tensor factorization, Proceedings of *Tenth IEEE International Conference on Computer Vision (ICCV)*, Beijing, China, 17-21 Oct. 2005.

[19] A.J. Ferreira and M.A.T. Figueiredo, On the use of independent component analysis for image compression, *Signal Processing: Image Communication*, 21(5):378–389, 2006.

[20] D.S. Taubman and M.W. Marcellin. *JPEG 2000: Image Compression Fundamentals, Standards and Practice*, Kluwer Academic Publishers, Norwell, MA, USA, 2001.

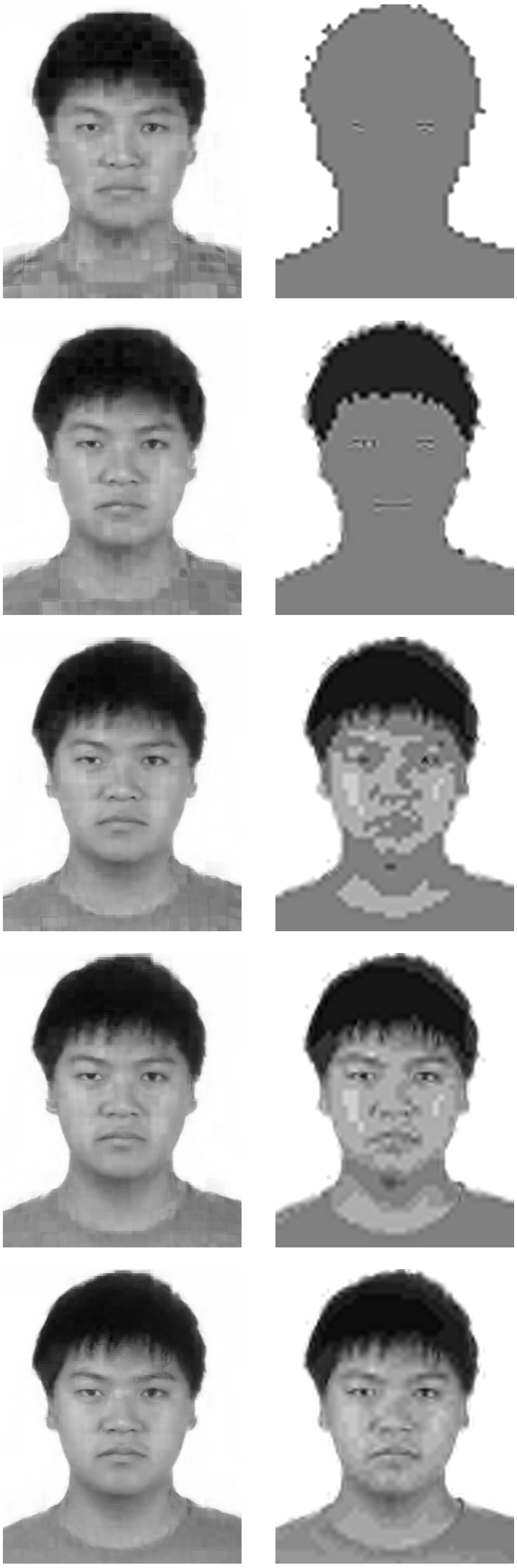
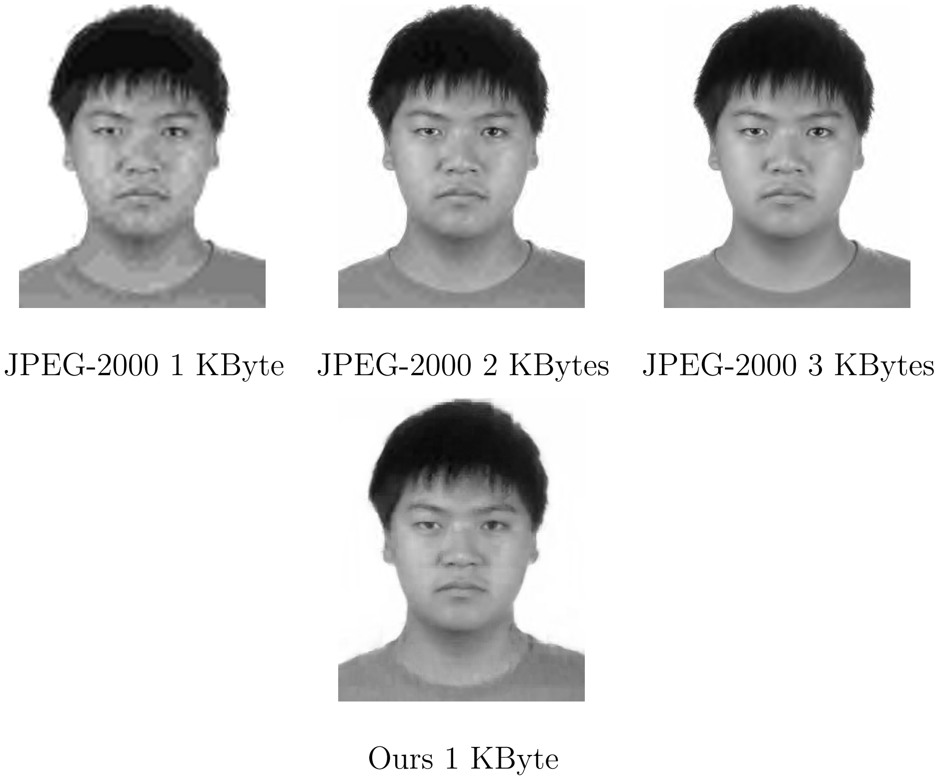


Fig. 8. Left - our results, Right - JPEG-2000. First row: 270 Bytes; Second row: 392 Bytes; Third row: 522 Bytes, Fourth row: 644 Bytes; and Last row: 865 Bytes.



JPEG-2000 1 KByte JPEG-2000 2 KBytes JPEG-2000 3 KBytes

Ours 1 KByte

Fig. 9. Our result (bottom) and JPEG-2000 results with higher bit-rates for comparison.

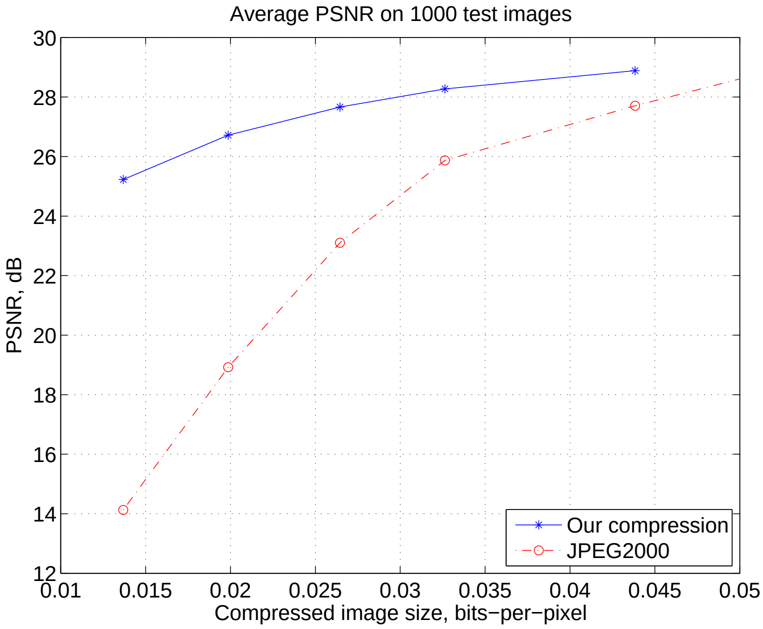


Fig. 10. Rate-distortion curves for JPEG-2000 and our results.