Appears in: Proceedings of International Workshop on Synthetic - Natural Hybrid Coding and Three Dimensional Imaging Rhodes, Greece, September 1997

Very Low Bit Rate Facial Texture Coding

J. Ström, F. Davoine, J. Ahlberg, H. Li and R. Forchheimer

Linköping University, 581 83 LINKÖPING, Sweden, Email: jacobs@isy.liu.se

ABSTRACT

This paper presents two methods of enhancing the eigen face approach for the compression of images of human faces. In earlier work, a series of image basis functions is created using the Karhunen-Loeve transform on a set of face images. An image can then be encoded as a linear combination of these basis functions. To increase efficiency, we propose a non-linear geometrical normalization of the images. To lower complexity, the images can be divided into blocks before the KL transform. An improvement on JPEG of 7 dB has been measured on individual images for low bit rates.

1. INTRODUCTION

An efficient way to encode images of human faces at low bit rates would be of great importance to the field of model based coding. If the face texture could be updated each frame, wrinkles and shadows generated from different expressions and lighting conditions could be compensated for. This will be too bit expensive to do with traditional waveform coders, a 200x200 pixel area of reasonable quality demands about 2Kbytes. This paper presents a coding scheme that aims at facial texture coding with resonable quality at bit rates of the same order of magnitude as the motion- and expression- data (roughly 500 bits per frame [1]). The next section presents previous work in this area, and the third section discusses non-convexity. The fourth section is about normalization, the encoding process is discussed in section five, and different ways to improve this process are covered in section six. The last section presents conclusions and future work.

2. BACKGROUND

When compressing a very clearly defined class of images such as facial texture images, all-purpose basis functions are obviously not the best choice. Moghaddam and Pentland [2] use the Karhunen-Loeve transform to create a set of basis functions that depend on the input class. A training set is built up by a number of images that are scaled to a fixed size. A KL transform is performed to generate a set of basis functions called eigen faces, which are combined linearly to approximate the image to encode.

3. CONVEXITY

If we represent each image with a vector, each image becomes a point in a high dimensional *image space*. The class of all face images would then form a data manifold called the *face space*, illustrated in figure 1 for 600x600 pixel images. It is easy to see that this data manifold is not

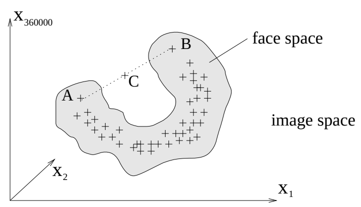

Figure 1: *What a face space could look like.*

convex; let point A in figure 1 represent the face of a very narrow eyed person, whereas B would represent the face of a person with a very long distance between his or her eyes. Now, if the face space would be convex, all convex combinations $\alpha A + (1-\alpha)B, \quad \alpha \in [0,1]$ would be inside the face space. But $C = 0.5A + 0.5B$ (depicted in figure 2), would result in an image with four eyes, which hardly can be thought of as a face, and must thus be outside the face space. Hence, the face space is not convex. It is clear that

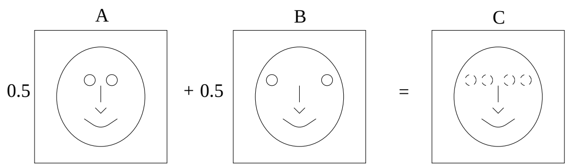

Figure 2: *The image C is not a face image.*

linear basis functions such as the ones achieved through the KL transform are sub-optimal. Either we will not be able to compress narrow eyed faces, or we will have an overcomplete base which can represent four-eyed creatures that do not exist. This problem will also occur when compressing both open and closed mouths, asymmetric faces, tilted eyes etc.

4. GEOMETRICAL NORMALIZATION

Since linear transforms are not adequate for non-convex sets, a preprocessing step of geometrical normalization is proposed in this paper. It is a non-linear transformation that will change the face space to a more convex shape, as depicted in the upper part of figure 3. The transformation process is illustrated in the bottom row. First the original face is segmented out from the background (first image). Then the CANDIDE model [3] is manually aligned to the face, as seen in the second image. The CANDIDE mesh is then warped to a standard geometry (third image), and the result is the normalized face (fourth image). In figure 4, three different faces are shown before (top row) and after (bottom row) geometrical normalization. Note that the faces are more similar to each other after the normalization process than before.

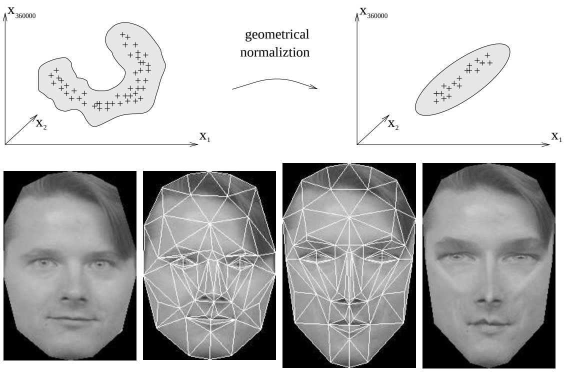

Figure 3: *Top: A non-linear transform that makes the face space more convex. Bottom, from left: Original image, original with aligned mesh, mesh warped to standard geometry and resulting image.*

5. THE ENCODING PROCESS

All N images in the training set are first geometrically normalized. In our experiment we used $N = 100$ images of size 600x600 pixels. The mean image of the N images is subtracted from the training set, and the KL transform is performed, yielding N eigen textures sorted in decreasing eigenvalue order. This set of eigen textures is called the *eigen set*. A new set called the *data base* is created from the first n eigen textures in the eigen set. The image to encode is first geometrically normalized and then projected onto the eigen textures in the data base. The face image is now represented by the n KL coefficients achieved in the projection and by the mesh data needed to reverse the geometrical normalization. An example is shown in figure 5. To the left, the original image (600x600 pixels) which was not part of the training set. After encoding and decoding, we get the right hand image. Without quantizing the coefficients, we would need 400 bytes for the coefficients and 195 bytes for the mesh data, for a PSNR of 27 dB. Quantizing the coefficients would lower the bit rate; one byte per coefficient should be sufficient. Further, in a model based coder, the mesh data (which is the head motion- and facial expression

Figure 4: *Top row: original images. Bottom row: geometrically normalized images.*

Figure 5: *Left: original image, (not part of the training set). Right: coded (n=N=100) and restored image.*

data) will already be sent once to the decoder, so the 100 additional bytes of KL coefficients would be sufficient for decoding the texture.

6. INCREASING THE QUALITY

A bit rate of 100 bytes per frame is of the same order of magnitude as the head motion- and facial expression data of 500 bits per frame, but a PSNR of 27 dB is not sufficient for all applications. Increasing the size of the training set would increase the quality, as seen in figure 6. The x-axis is the size of the data base, and the

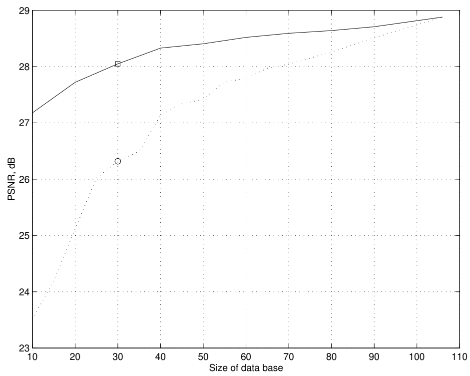

Figure 6: *Dotted line: Size of training set = size of data base. Solid line: Size of training set = 106.*

y-axis is the measured PSNR between the normalized images before and after coding. For the dotted curve, the size of the training set and the size of the data base are kept equal ($n = N$), and the PSNR increases as $n = N$ goes from 10 to 106. For the solid curve, the size of the training set (N) is 106 all the way, while n (the size of the data base) goes from 10 to 106. Note that the big training set pays off already for small values of n. For instance, for a data base of 30 images, we gain 1.7 dB by increasing the size of the training set from 30 to 106 (marked with a circle and a square respectively in figure 6). Once the data base is built the size of the training set no longer influences the complexity nor the storage demands of the algorithm. For a fixed size of the data base, it would thus make sense to increase the size of the training set until no further significant quality gain is achieved.

6.1. Enhancing the training set

One approach to increase the size of the training set without adding new faces (which are normalized manually as for now) would be to mirror the data, as illustrated in figure 7. For a fixed training set of size $N = 50$, the solid curve shows

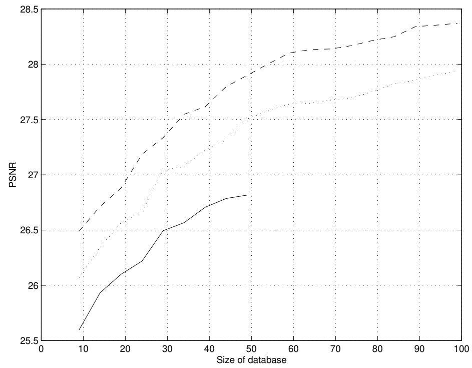

Figure 7: *Mirroring the training set. Solid curve: A training set of 50 images. Dotted curve: The same training set, mirrored to the size of 100. Dashed curve: Training set of 100 images.*

how the PSNR goes up with an increasing size of the data base (n). The training set is then enhanced with 50 new images ($N = 100$) and the dashed curve is generated with n going from 10 to 100. As expected, this curve is always better in PSNR than the solid one. For the dotted curve, we mirror the 50 images to produce a training set of size $N = 100$. This curve seems to go right in the middle between the two others. Thus the gain of mirroring does not seem to be insignificant.

6.2. Blurring the training set

Reasonings, for which there is not enough space in this paper to describe, led us to investigate whether blurring the training set could increase performance. In figure 8, the result of such an experiment is presented. The solid curve is gen-

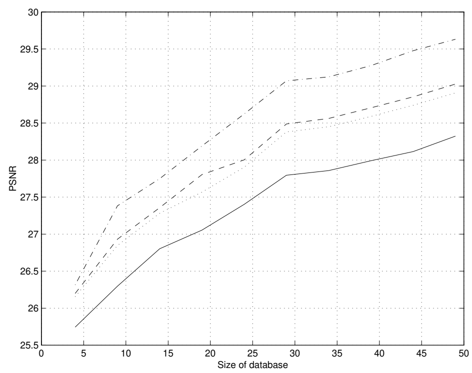

Figure 8: *Blurring the training set. Solid curve: Original training set. Dotted, dashed and dot-dashed curve: Training set blurred with a Hamming filter of size 2, 4 and 8 respectively.*

erated from a training set of 50 images without blurring. The dotted, dashed, and dashed-dotted curves corresponds to training sets that are blurred with a Hamming window of size two, four and eight respectively. As seen in the diagram, the PSNR increases the more blurred the image get. However, at a Hamming window size of eight, the blurring becomes so evident in the decoded image that it is visually worse than the non-blurred version.

6.3. Partitioning the image into blocks

When increasing the size of the data base, complexity problems arise. For instance, a data base of 5000 eigen textures of 600x600 pixels would demand a storage capacity of 1.7 Gbytes, and encoding (or decoding) 25 images per second would require a computing power of 41 Gflops. To enhance quality without increasing complexity, we divide the normalized image into blocks. For each block, an eigen space is created from the all the blocks in the training set of that position. Such a modular eigen space description of the face was first presented by Pentland [4] for the purpose of image recognition. In our algorithm, a block size is 32x32 pixels is used, and a training set of 100 images mirrored to 200 is used without blurring the images. The data base is of size 13 for all blocks, thereby reducing complexity by a factor 8 compared to the full face version of figure 5, and the coefficients are quantized with a scalar quantizer to 5 bits per coefficient. This gives a total of 1632 bytes, including the mesh information. The result is presented in figure 9. Figure 9a is the original image (not part of the training set), 9b is coded with our block based algorithm presented above, yielding a PSNR of 35.5 dB. Figure 9c is coded with JPEG to a size of 2508 bytes, with a PSNR of 26.0 dB. Since this is not optimal operational conditions for JPEG, we also tried to subsample the image before it was JPEG coded (9d), yielding 1675 bytes and a PSNR of 28.9 dB.

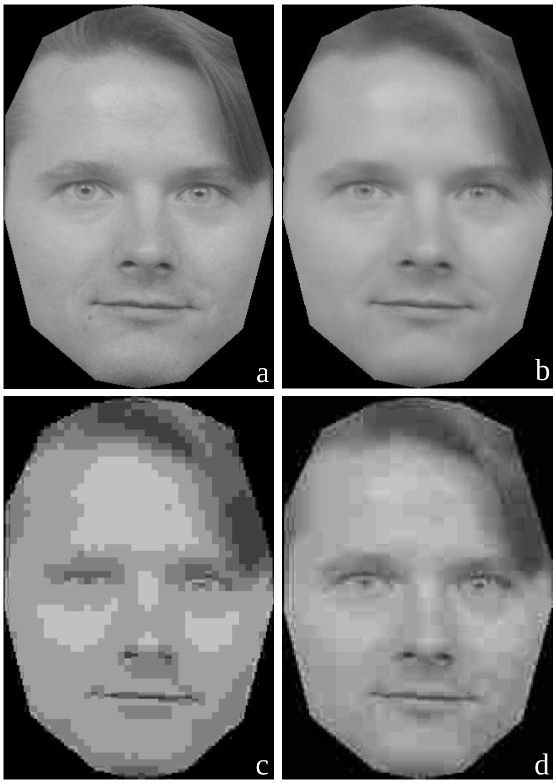

Figure 9: *(a) Original image, not included in the training set. (b) Our algorithm, 35.5 dB (c) JPEG, 26.0 dB. (d) Subsampled JPEG, 28.9 dB. Images of size 1632, 2508 and 1675 bytes respectively.*

7. CONCLUSIONS AND FUTURE WORK

In this paper we have suggested to enhance the eigen face compression approach with a non-linear geometrical normalization step. This is done to make the face space more convex, and thus better suited for the linear KL transform. We also investigate different methods for enhancing the training set without adding new face images; mirroring and blurring seem to improve performance. Moreover, we propose a method to lower the complexity by dividing the images into blocks before the KL transform. An improvement on JPEG of 7 dB is measured at low bit rates. Future work is needed to figure out how to best quantize the coefficients in the block based scheme. Automating the geometrical normalization step is very essential for the feasibility of this entire scheme.

8. REFERENCES

[1] H. Li, A. Lundmark and R. Forchheimer, "Image sequence coding at very low bit rates: A review,", *IEEE Trans. Image Proc.*, vol. 3, pp. 589–609, Sept 1994.

[2] Baback Moghaddam and Alex Pentland, "An Automatic System for Model-Based Coding of Faces" *M.I.T. Media Laboratory Perceptual Computing Section Technical Report No. 317*

[3] M. RydFalk, "CANDIDE, a parameterized face", *Technical Report, LiTH-ISY-I-0866, Linköping University*, 1987.

[4] A. Pentland, B. Moghaddam and T. Starner, "View-Based and Modular Eigenspaces for Face Recognition", *Proceedings 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 21-23 June 1994.