IMPROVING THE K-SVD FACIAL IMAGE COMPRESSION USING A LINEAR DEBLOCKING METHOD

Ori Bryt

The Electrical Engineering Department
The Technion - Israel Institute of Technology
stinger@cs.technion.ac.il

Michael Elad

The Computer Science Department
The Technion - Israel Institute of Technology
elad@cs.technion.ac.il

ABSTRACT

The use of sparse representations in signal processing is gradually increasing in the past several years. In a previous work we proposed a new method for compressing facial images using the K-SVD algorithm, which is a novel algorithm for training overcomplete dictionaries that lead to sparse signal representations. This method was shown to be most efficient, surpassing the JPEG2000 performance significantly.
In this paper we present a significant addition to our compression algorithm in the form of image deblocking. Since the encoding is done in patches, a visually disturbing artifacts of blockiness appear in the reconstructed images. We eliminate these artifacts using a linear deblocking technique, which is based on local image filters. We construct a linear filter for each relevant pixel independently, and apply these filters as post-processing. This method is limited, but nevertheless it improves the PSNR of the reconstructed images and gives visually appealing results.

Index Terms— Image compression, Facial images, Sparse representations, K-SVD, Deblocking.

1. INTRODUCTION

Compression of still images exploits their vast spatial redundancy and the ability to absorb moderate errors in the reconstructed image. When considering the compression of a specific and narrow family of images, the redundancy even increases, thus enabling a better compression performance. Such is the case with compression of facial images.

Compression of facial images is an appealing problem, both in research and in applications. From a research perspective, one should primarily wonder how to exploit the additional redundancy that such a focused family of images exhibits, and how to surpass general purpose compression algorithms this way. Application-wise, facial images are perhaps the most popular images, held in large databases by various organizations, and efficient storage of such images is of value.

Motivated by the above, the work reported in [1] address compression of facial images as well, targeting compression ratios of 100 and beyond, where most algorithms are unable to show recognizable faces. The developed algorithm relies strongly on advancements made in using sparse and redundant representation of signals [2, 3], and learning their sparsifying dictionaries [4, 5, 6]. The K-SVD algorithm is a novel method for training overcomplete dictionaries that lead to sparse signal representations. It is used in [1] for learning the dictionaries for representing small image patches in a locally adaptive way, and use these to sparse-code the patches' content. This is a relatively simple and straight-forward algorithm with hardly any entropy coding stage. Yet, it is shown to be superior to several competing algorithms, such as JPEG, JEPG2000, local-PCA and VQ coding.

In spite of the superior performance over these algorithms, there is still room for improvement in the quality of our reconstructed images. One of the more apparent problems is the one of blockiness, which is created due to the method of encoding - independently in uniform non-overlapping patches. Neighboring patches aren't fitted to match each other, and thus false contours occur between them. In this work, which is a direct continuation to [1], we attempt to improve the quality of the reconstructed images by applying a linear post-processing method of deblocking, based on Least Squares (LS). This deblocking method is unique, fast and efficient, and improves the visual quality of the images and their PSNR.

This paper is organized as follows: In section 2 we provide some background material on sparse representation, dictionary learning and the coding algorithm devised in [1] on which we build here. In Section 3 we present out deblocking method, and in section 4 we demonstrate its behavior.

2. PREVIOUS WORK

2.1. Background on Sparse Representations

We begin by describing the model for signals used in the compression algorithm presented in [1]. We consider a family of image patches of size $N \times N$ pixels, ordered lexicographically as column vectors $\mathbf{x} \in \mathbb{R}^n$ (with $n = N^2$). Assume that we are given a matrix $\mathbf{D} \in \mathbb{R}^{n \times k}$ (with possibly $k > n$). We refer

hereafter to this matrix as the dictionary. The proposed model suggests that every such image patch, $\mathbf{x}$, could be represented sparsely using this dictionary, i.e., the solution of

$$\hat{\alpha} = \underset{\alpha}{\operatorname{argmin}} \ \|\alpha\|_0 \ \text{ subject to } \ \|\mathbf{D}\alpha - \mathbf{x}\|_2^2 \leq \varepsilon^2 \qquad (1)$$

is expected to be very sparse, $\|\hat{\alpha}\|_0 \ll n$. The notation $\|\alpha\|_0$ counts the non-zero entries in α. Thus, every signal instance from the family we consider is assumed to be represented as a linear combination of a few columns (referred to hereafter as *atoms*) from the redundant dictionary $\mathbf{D}$.

The requirement $\|\mathbf{D}\alpha - \mathbf{x}\|_2 \leq \varepsilon$ suggests that the approximation of $\mathbf{x}$ using $\mathbf{D}\alpha$ need not be exact, and could absorb a moderate error ε. This suggests an approximation that trades-off accuracy of representation with its simplicity. Indeed, compression of $\mathbf{x}$ can be achieved by transmission of the vector $\hat{\alpha}$, by specifying the indices of its non-zero elements and their magnitudes. Quantization on the non-zero entries in $\hat{\alpha}$ leads to higher approximation error but the transmission of $\hat{\alpha}$ now requires a finite and small number of bits.

In order to use this model for compression of image patches, we need an effective way to solve the problem posed in Equation (1). While this problem is in general very hard to solve, the matching and the basis pursuit algorithms can be used quite effectively [2, 3] to get an approximated solution. We refer hereafter to Equation (1) as (P_0) and to its solution as *sparse coding*.

So far we have assumed that the dictionary $\mathbf{D}$ is known. In practice, this is achieved by training $\mathbf{D}$ – minimizing the following energy functional with respect to both $\mathbf{D}$ and $\{\alpha_j\}_{j=1}^M$.

$$\varepsilon\left(\mathbf{D}, \{\alpha_j\}_{j=1}^M\right) = \sum_{j=1}^{M} \left[\mu_j \|\alpha_j\|_0 + \|\mathbf{D}\alpha_j - \mathbf{x}_j\|_2^2\right] \qquad (2)$$

This expression seeks to get a sparse representation per each of the examples in $\mathcal{X}$, and obtain a small representation error. The choice for μ_j dictates how those two forces should be weighted, so as to make one of them a clear constraint. For example, constraining $\forall\ j\ \ \|\alpha_j\|_0 = L$ implies specific values for μ_j, while requiring $\forall\ j\ \|\mathbf{D}\alpha_j - \mathbf{z}_j\|_2 \leq \varepsilon$ leads to others. The K-SVD algorithm proposes an iterative algorithm designed to handle the above task effectively [5]. Adopting a block-coordinate descent idea, the computations of $\mathbf{D}$ and $\{\alpha_j\}_{j=1}^M$ are separated. Assuming that $\mathbf{D}$ is known, the penalty posed in Equation (2) reduces to a set of M sparse coding operations. Similarly, assuming that these representation vectors are fixed, the K-SVD algorithm proposes an update of the dictionary one column at a time.

2.2. The Compression Algorithm

The compression algorithm presented in [1] consists of two main processes: An off-line K-SVD training and an on-line image compression and decompression. The K-SVD training process is preformed off-line, preceding any image compression. It produces a set of K-SVD dictionaries that are then considered fixed for the image compression stage. A single dictionary is trained for each patch over training examples. Prior to the coding stage, a geometrical warp as in [7] is applied for aligning the images. This leads to better conditioning of the input image, increasing its redundancy versus the image database. The warp parameters are coded using ~ 20 bytes. Pre-processing also includes a scale-down by $2:1$ of the input image. Both actions are reversed in post-processing.

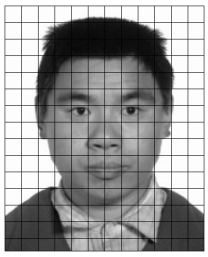

Fig. 1. A uniform slicing to disjoint square patches.

In the actual coding stage the image is divided into square patches of fixed size, coding each separately by sparse-coding it. In [1], patches of size $N = 15$ pixels are used as in Figure 1, although larger patches are also possible. These patches fit their relative dictionaries in content. Every patch has a pre-trained dictionary of atoms $\mathbf{D}_{ij}$. The dictionary size is $k = 512$. The coding is done by assigning a linear combination of a few atoms to describe the patch's content. Thus, information about the content includes both the linear weights and the involved indices, and both the encoder and the decoder are storing the dictionaries. The sparse coding for encoding of the patches is done using the OMP algorithm because of its simplicity and efficiency. The decoder is simply aggregating the proper atoms with the proper weights to reconstruct the patch, building the output one patch at a time independently of the others.

3. THE DEBLOCKING METHOD

3.1. The Blockiness Artifacts

The motivation for the deblocking method is the appearance of blockiness artifacts in the reconstructed images that result from our compression algorithm. These disturbing artifacts are visible all over the image in varying degrees of strength, and all the reconstructed images suffer from them. The blockiness artifacts are created because the coding of each image is done in uniform non-overlapping patches, as can be seen in Figure 1, and these patches are coded independently according to their relevant trained dictionaries. Therefore, the nature of the representation in each patch is different, and especially so in patches which require a different number of atoms or which have a different pattern of data in them. These dif-

ferences can create false edges between the patches, as can be seen in the top row of Figure 4. This phenomenon is expected to appear in every compression technique in which the encoding is done in non-overlapping blocks, but it is more significant in our method, due to the deep compression ratio, and the size of the blocks. Naturally, block sizes could be decreased and dictionaries could be extended, but doing so will not eliminate the artifacts completely and will severely hurt the performance of our compression algorithm as the rate of each compressed image will increase. Such a solution is thus inefficient from a wider point of view.

3.2. Possible Approaches For Solution

There are several possible approaches to eliminate the blockiness artifacts, which mostly divide into two main categories: Encoding stage approaches and post-processing approaches. The encoding stage approaches suggest attacking the problem in its very basis, preventing the blockiness from being created by altering the encoding stage to support a new model, one in which blockiness does not appear in the reconstructed images. Such possible approaches include encoding in overlapping blocks, train dictionaries on two or more block grids, train dictionaries with respect to their neighboring dictionaries and other approaches along that line. These approaches offer a clean structural solution to the problem and are designed to eliminate it completely. However, the disadvantages of these approaches are substantial - they are complex, some are complicated to implement and they can lead to an increase in the rate of the compressed images, which is undesirable.

We made an attempt to implement an encoding stage which uses overlapping blocks. It brought to an increase in the number of blocks, which naturally increased the rate of the compressed images without significant improvement to the visual quality. Thus, the results were even worse per each rate. A further increase in the overlapping will further increase the rate and the improvement in the quality is unlikely to compensate for it.

In contrary to encoding stage approaches, post-processing approaches attempt to eliminate the blockiness artifacts after they have already been created by the compression algorithm. Such possible approaches include all sort of filtering techniques applied to an image. These approaches are simple and easy to implement, and most of them do not increase the rate of the compressed image. The main disadvantage of these methods is of course that they cannot guaranty to eliminate the blockiness artifacts in all cases and all images, as the method of operation is limited and does not have a direct relation to the source of the problem.

3.3. Our Deblocking Approach

Our deblocking method belongs to the second category, as it is an application of specially created filters over the reconstructed images. We define a three pixel wide border-frame of each block as our region of interest to be fixed. These areas suffer the most from the blockiness artifacts. For each pixel in the region of interest of each different block we calculate *a different* 5*x*5 linear filter to correct the blockiness artifacts according to a learning group of the same pixel in similar images. The learning group is the source for creating a model for these artifacts by using both the original versions of the images and the reconstructed ones. In general, this approach is similar to the principles of our compression algorithm, in which the encoding stage for new images is done according to dictionaries that were trained according to a learning group. In this case, however, we do not train the linear filters according to the learning group, but rather calculate them from the following equation:

$$\mathbf{h_k} = \underset{\mathbf{h_k}}{\operatorname{argmin}} \ \|\mathbf{Y}\mathbf{R_k}\mathbf{h_k} - \mathbf{X}\mathbf{e_k}\|_2^2, \tag{3}$$

where $\mathbf{h_k}$ is the desired 5×5 filter for the pixel $\mathbf{k}$, ordered as a column vector. The matrix $\mathbf{X}$ contains the original images in the learning group as its rows, and similarly, $\mathbf{Y}$ is the matrix containing the decoded images in the learning group as rows. $\mathbf{R_k}$ is a matrix that reduces $\mathbf{Y}$ to the appropriate pixels in the 5*x*5 environment of the $\mathbf{k}$-th pixel, and $\mathbf{e_k}$ is a vector that selects the $\mathbf{k}$-th pixel out of the images $\mathbf{X}$.

The goal of this minimization problem is to create a linear filter that attempts to fit the pixel $\mathbf{k}$ in the reconstructed image to the pixel $\mathbf{k}$ in the original image in the best possible way, using only the pixel's 5*x*5 surrounding. Equation 3 has a simple analytic solution for $\mathbf{h_k}$ in the form of the following Least Squares equation:

$$\mathbf{h_k} = (\mathbf{R_k^T}\mathbf{Y^T}\mathbf{Y}\mathbf{R_k})^{-1} \cdot \mathbf{R_k^T}\mathbf{Y^T}\mathbf{X}\mathbf{e_k}. \tag{4}$$

We chose a set of rates as working points and calculated a filter for each pixel in the region of interest of each block for each of these rates. For each rate we combined the appropriate filters as rows to a matrix $\mathbf{H_r}$ in their matching places, filling the other rows with appropriate rows from the identity matrix of the same size.

The filter calculation process is simple, easy to implement and does not increase the rate of the compression, as the resulting matrices of filters for each rate are calculated in off-line mode and are stored in the decoder. The method of using these filters on any given reconstructed image is also very convenient, simply by multiplying a filters matrix $\mathbf{H_r}$ by the column-stacked representation of the image in the post-processing stage of our compression algorithm.

Although it has many advantages, this method is limited in its ability to eliminate blockiness artifacts since it uses only a relatively small linear filter in each pixel when trying to create a model for solving the problem. Thus, the model that can be created by this method is simple, while the nature of the problem is most likely not, and could even be image dependent.

Fig. 2. A comparison of facial images compression with a bit-rate of 550 bytes without deblocking. The values in the brackets show the representation RMSE.

4. RESULTS

We used $4,415$ facial images as our learning set and 100 different images as our test set. All the images in both sets are of size 179×221 after pre-processing. First, for completeness of the presentation, we present the compression algorithm results, and their comparison to several alternative methods, JPEG, JPEG2000, the VQ-Based compression algorithm [7], and a PCA-Based compression algorithm described in [1]. Figure 2 shows a visual comparison of our results without deblocking with the JPEG, JPEG2000, and PCA for a bit-rate of 550 bytes. This figure clearly show the visual and quantitative advantage of our method over all the alternatives. We can specially notice the strong smearing effect in the JPEG and JPEG2000 images, whereas our images are clear and sharp. A similar behavior was observed in other bit-rates [1].

Figure 3 shows a Rate-Distortion curves comparison of the compression methods mentioned before, averaged on the test set. The PSNR shown here corresponds to the aligned grayscale images for all methods, in order to evaluate the representation error from the sparse coding and reconstruction stages alone, without the error that result from the geometrical warp process. Adding the error induced by the geometrical warp implies a decrease of $0.2-0.4$dB for the VQ, PCA, and K-SVD methods. Explanation about the "clean JPEG2000" can be found in [1].

We now turn to discuss the deblocking method and its results. In order to show the effectiveness of our deblocking method we used it on the same images as in figure 2 in the same rate of 550 bytes. We also added one additional image where the blockiness artifacts were particulary strong. The filters matrix that was calculated for this rate approximately was used on the reconstructed images as a post-processing operation and the results can be seen in Figure 4. In spite it its limitations, it can be clearly seen that the deblocking method has eliminated the blockiness artifacts in most places and reduced them significantly in others. Although the deblocking filters create a small degree of smearing along the block edges where they are applied, the images still appear sharp and their overall visual quality is improved. In addition to the improvement in the visual quality, which was our primary goal in creating this method, we also gain an improvement in the PSNR of the images [1] and thus extend our lead in performance over the other methods, and especially JPEG2000.

5. CONCLUSION

In a previous paper, we presented a novel and highly efficient facial image compression algorithm, based on sparse and redundant representations and the K-SVD algorithm. In this paper we introduce an important post-processing addition of image deblocking to it. The proposed deblocking is tested in various bit-rates and options with great success. The additional deblocking method is a fundamental method that can fit with many other block-based compression techniques. It improves over our previous results and create a more appealing and smooth visual images.

6. REFERENCES

[1] O. Bryt and M. Elad, "Compression of Facial Images Using the K-SVD Algorithm", *Journal of Visual Communication and Image Representation*, Vol. 19(4), pp. 270–283, 2008.

[2] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries", *IEEE Trans. Signal Processing*, Vol. 41(12), pp. 3397–3415, 1993.

[3] S.S. Chen, D.L. Donoho, and M.A. Saunders, "Atomic decomposition by basis pursuit", *SIAM Rev.*, Vol. 43(1), pp. 129–159, 2001.

[4] M. Aharon, M. Elad, and A.M. Bruckstein, "On the uniqueness of overcomplete dictionaries, and a practical way to retrieve them", *Journal of Linear Algebra and Applications*, Vol. 416, pp. 48–67, 2006.

[5] M. Aharon, M. Elad, and A.M. Bruckstein, "The K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representation", *IEEE Trans. on Signal Processing*, Vol. 54(11), pp. 4311–4322, 2006.

[6] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries", *IEEE Trans. on Image Processing*, Vol. 15(12), pp. 3736–3745, 2006.

[7] M. Elad, R. Goldenberg, and R. Kimmel, "Low bit-rate compression of facial images", *IEEE Trans. on Image Processing*, Vol. 16(9), pp. 2379–2383, 2007.

[1] On average. Some images suffer a small decrease in their PSNR.

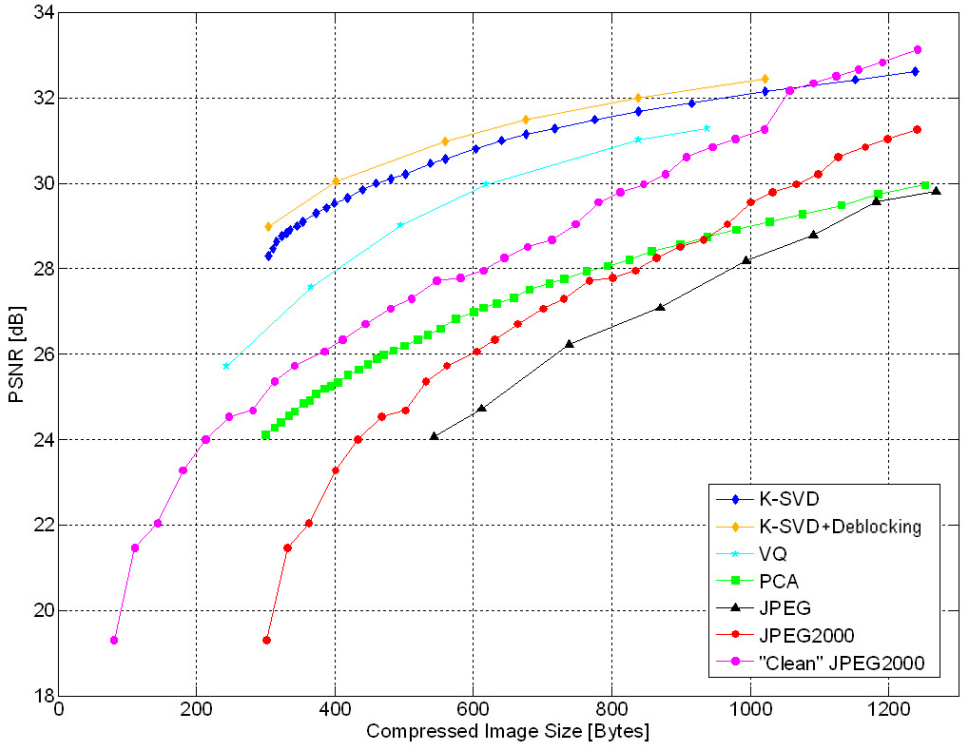

Fig. 3. Rate-Distortion curves for the JPEG, JPEG2000, ”Clean” JPEG2000, PCA, VQ, and the two K-SVD methods.

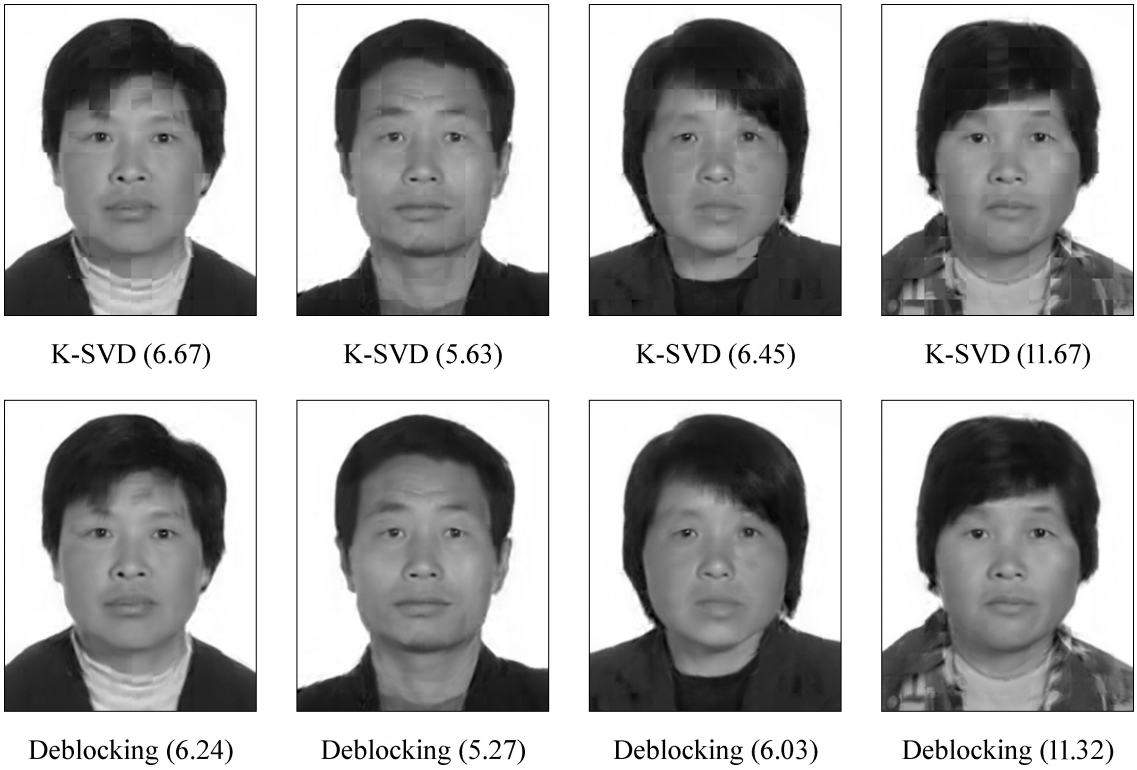

Fig. 4. A comparison of reconstructed images in the rate of 550 bytes before (top) and after (bottom) applying the linear deblocking method.