J. Vis. Commun. Image R. 19 (2008) 270–282

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

Compression of facial images using the K-SVD algorithm ☆

Ori Bryt [a], Michael Elad [b,*]

[a] The Electrical Engineering Department, The Technion—Israel Institute of Technology, Technion City, Haifa 32000, Israel
[b] The Computer Science Department, The Technion—Israel Institute of Technology, Taub Building, Office 516, Technion City, Haifa 32000, Israel

A R T I C L E   I N F O

Article history:
Received 3 November 2007
Accepted 7 March 2008
Available online 16 March 2008

Keywords:
Image compression
Sparse representations
Redundancy
K-SVD
OMP
Facial images
PCA
JPEG
JPEG2000
VQ

A B S T R A C T

The use of sparse representations in signal and image processing is gradually increasing in the past several years. Obtaining an overcomplete dictionary from a set of signals allows us to represent them as a sparse linear combination of dictionary atoms. Pursuit algorithms are then used for signal decomposition. A recent work introduced the K-SVD algorithm, which is a novel method for training overcomplete dictionaries that lead to sparse signal representation. In this work we propose a new method for compressing facial images, based on the K-SVD algorithm. We train K-SVD dictionaries for predefined image patches, and compress each new image according to these dictionaries. The encoding is based on sparse coding of each image patch using the relevant trained dictionary, and the decoding is a simple reconstruction of the patches by linear combination of atoms. An essential pre-process stage for this method is an image alignment procedure, where several facial features are detected and geometrically warped into a canonical spatial location. We present this new method, analyze its results and compare it to several competing compression techniques.

1. Introduction

Compression of still images is a very active and matured field of research, vivid in both research and engineering communities. Compression of images is possible because of their vast spatial redundancy and the ability to absorb moderate errors in the reconstructed image. This field of work offers many contributions, some of which became standard algorithms that are wide-spread and popular. Among the many methods for image compression, one of the best is the JPEG2000 standard—a general purpose wavelet based image compression algorithm with very good compression performance [1].

When considering the compression of a specific and narrow family of images, the above-mentioned redundancy increases, thus enabling a better compression performance. Such is the case with compression of facial images. Indeed, this expected gain has been observed and exploited in several recent publications that offer tailored compression algorithms for facial images [2–17]. Among those contributions, the more recent ones show performance surpassing those of the JPEG2000 [16,17].

Compression of facial images is an appealing problem, both because of the research challenges it provides, and the important applications it serves. 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. This is not an easy challenge due to the vast efforts put to the general purpose algorithms, and especially their entropy coding parts.

Application-wise, facial images are perhaps the most popular images, held in large databases by various organizations, such as police and law-enforcement, schools and universities, states, and in databases of employees in large companies. Efficient storage of such images is of value, and especially so when considering photo-ID in an electronic ID cards. In such applications, very-low bit-rate compression is to be considered, where general purpose algorithms fail utterly.

Motivated by the above, In this work we address compression of facial images as well, very much in line with the above activity. In order to focus the discussion, we target photo-ID images of pre-specified and fixed size 358×441 grayscale images with 8 bits per pixel.[1] The goal of our work is very-low bit-rates (compression ratios of 200 and beyond), where most algorithms are unable to

☆ This research was partly supported by the Israel Science Foundation Grant No. 796/05.

* Corresponding author.
E-mail addresses: stinger@cs.technion.ac.il (O. Bryt), elad@cs.technion.ac.il (M. Elad).

[1] This database contains color photos taken by a 4-Mega-pixel digital camera (Fuji FinePix A400), against a white and uniform background, with the highest compression settings for best quality. These photos were cropped and scaled to the above-mentioned size, and also converted to a gray-value format.

1047-3203/$ - see front matter © 2008 Published by Elsevier Inc.
doi:10.1016/j.jvcir.2008.03.001

show recognizable faces. We use a database containing around 6000 such facial images, some of which are used for training and tuning the algorithm, and the others for testing it, similar to the approach taken in [17].

In our work we propose a novel compression algorithm, related to the one presented in [17], improving over it.

Our algorithm relies strongly on recent advancements made in using sparse and redundant representation of signals [18–26], and learning their sparsifying dictionaries [27–29]. We use the K-SVD algorithm 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: (i) the JPEG2000, (ii) the VQ-based algorithm presented in [17], and (iii) A Principal Component Analysis (PCA) approach.[2]

In the next section we provide some background material for this work: we start by presenting the details of the compression algorithm developed in [17], as their scheme is the one we embark from in the development of ours. We also describe the topic of sparse and redundant representations and the K-SVD, that are the foundations for our algorithm. In Section 3 we turn to present the proposed algorithm in details, showing its various steps, and discussing its computational/memory complexities. Section 4 presents results of our method, demonstrating the claimed superiority. We conclude in Section 5 with a list of future activities that can further improve over the proposed scheme.

2. Background material

2.1. VQ-based image compression

Among the thousands of papers that study still image compression algorithms, there are relatively few that consider the treatment of facial images [2–17]. Among those, the most recent and the best performing algorithm is the one reported in [17]. That paper also provides a thorough literature survey that compares the various methods and discusses similarities and differences between them. Therefore, rather than repeating such a survey here, we refer the interested reader to [17]. In this sub-section we concentrate on the description of the algorithm in [17] as our method resembles it to some extent.

This algorithm, like some others before it, starts with a geometrical alignment of the input image, so that the main features (ears, nose, mouth, hair-line, etc.) are aligned with those of a database of pre-aligned facial images. Such alignment increases further the redundancy in the handled image, due to its high cross similarity to the database. The warping in [17] is done by an automatic detection of 13 feature points on the face, and moving them to pre-determined canonical locations. These points define a slicing of the input image into disjoint and covering set of triangles, each exhibiting an affine warp, being a function of the motion of its three vertices. Side information on these 13 feature locations enables a reverse warp of the reconstructed image in the decoder. Fig. 1 (left side) shows the features and the induced triangles. After the warping, the image is sliced into square and non-overlapping patches (of size 8×8 pixels), each of which is coded separately. Such possible slicing (for illustration purpose we show this slicing with larger patches) is shown in Fig. 1 (right side).

Coding of the image patches in [17] is done using vector quantization (VQ) [30–32]. The VQ dictionaries are trained (using tree-

[2] The PCA algorithm is developed in this work as a competitive benchmark, and while it is generally performing very well, it is inferior to the main algorithm presented in this work.

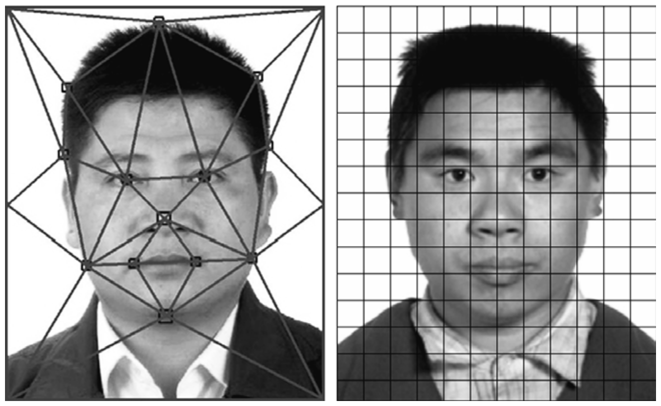
Fig. 1. (Left) Piece-wise affine warping of the image by triangulation. (Right) A uniform slicing to disjoint square patches for coding purposes.

K-Means) per each patch separately, using patches taken from the same location from 5000 training images. This way, each VQ is adapted to the expected local content, and thus the high performance presented by this algorithm. The number of code-words in the VQ is a function of the bit-allocation for the patches. As we argue in the next section, VQ coding is limited by the available number of examples and the desired rate, forcing relatively small patch sizes. This, in turn, leads to a loss of some redundancy between adjacent patches, and thus loss of potential compression.

Another ingredient in this algorithm that partly compensates for the above-described shortcoming is a multi-scale coding scheme. The image is scaled down and VQ-coded using patches of size 8×8. Then it is interpolated back to the original resolution, and the residual is coded using VQ on 8×8 pixel patches once again. This method can be applied on a Laplacian pyramid of the original (warped) image with several scales [33].

As already mentioned above, the results shown in [17] surpass those obtained by JPEG2000, both visually and in Peak-Signal-to-Noise Ratio (PSNR) quantitative comparisons. In our work we propose to replace the coding stage from VQ to sparse and redundant representations—this leads us to the next subsection, were we describe the principles behind this coding strategy.

2.2. Sparse and redundant representations

We now turn to describe a model for signals known as *Sparseland* [29]. This model suggests a parametric description of signal sources in a way that adapts to their true nature. This model will be harnessed in this work to provide the coding mechanism for the image patches. 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 *Sparseland* model suggests that every such image patch, $\mathbf{x}$, could be represented sparsely using this dictionary, i.e., the solution of

$$\hat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}} \|\boldsymbol{\alpha}\|_0 \text{ subject to } \|\mathbf{D}\boldsymbol{\alpha} - \mathbf{x}\|_2^2 \leqslant \varepsilon^2, \tag{1}$$

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

The requirement $\|\mathbf{D}\boldsymbol{\alpha} - \mathbf{x}\|_2 \leqslant \varepsilon$ suggests that the approximation of $\mathbf{x}$ using $\mathbf{D}\boldsymbol{\alpha}$ need not be exact, and could absorb a moderate error ε. This suggests an approximation that trades-off accuracy of representation with its simplicity, very much like the rate-distortion curve in compression. 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 with the blessed fact that transmission of $\hat{\alpha}$ now requires a finite and small number of bits. This is exactly the strategy we are about to deploy. Similar approach in a different context has been practiced in [34] for coding of general images.

In order to use the *Sparseland* model for compression of image patches, we need an effective way to solve the problem posed in Eq. (1). While this problem is in general very hard to solve, the matching and the basis pursuit algorithms can be used quite effectively [18,19,22,23] to get an approximated solution. Recent work established that those approximation techniques can be quite accurate if the solution is sparse enough to begin with [20,21,24–26]. In this work we make use of the Orthogonal Matching Pursuit (OMP) because of its simplicity and efficiency. We refer hereafter to the solution of (P_0) as *sparse coding*.

2.3. Training A dictionary

Given a set of image patches of size $N \times N$ to be coded, $\mathcal{X} = \{\mathbf{x}_j\}_{j=1}^M$, the assumption that they emerge from the *Sparseland* model helps us in devising the new coding strategy. However, we must have the dictionary $\mathbf{D}$ in order to use this coding scheme. This can be achieved by training $\mathbf{D}$—minimizing the following energy functional with respect to both $\mathbf{D}$ and $\{\alpha_j\}_{j=1}^M$,

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

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 $\forall j \|\mathbf{D}\alpha_j - \mathbf{z}_j\|_2 \leqslant \varepsilon$ leads to others.

The K-SVD algorithm proposes an iterative algorithm designed to handle the above task effectively [27,28]. 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 Eq. (2) reduces to a set of M sparse coding operations, very much like in Eq. (1). Thus, OMP can be used to obtain the near-optimal solutions. Similarly, assuming that these representation vectors are fixed, the K-SVD algorithm proposes an update of the dictionary one column at a time. As it turns out, this update can be done optimally, leading to the need to perform a singular value decomposition (SVD) operation on residual data matrices, computed only on the examples that use this atom. This way, the value of $\varepsilon(\mathbf{D}, \{\alpha_j\}_{j=1}^M)$ is guaranteed to drop per an update of each dictionary atom, and along with this update, the representation coefficients change as well (see [27,28] for more details).

Fig. 2 shows a typical representation error reduction graph in a K-SVD dictionary training process performing a gradual increment of the number of atoms used for the representation of the examples. This error is shown as Root Mean Squared Error (RMSE), being the squared-root of the mean of the errors $\|\mathbf{D}\alpha_j - \mathbf{x}_j\|_2^2$, i.e., setting $\forall j \|\alpha_j\|_0 = L$, the algorithm starts with $L = 1$ and after obtaining convergence, L is incremented by 1.

3. The proposed method

3.1. The general scheme

We now turn to describe our compression scheme. A block diagram of the encoder and decoder are given in Fig. 3. Our algorithm

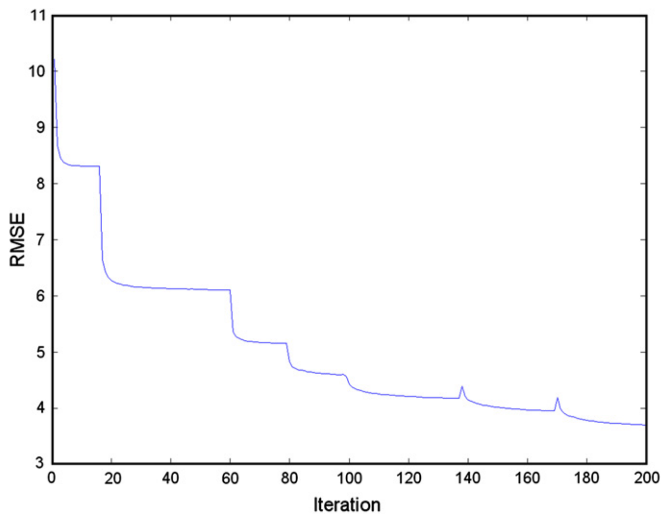

Fig. 2. A typical representation error (RMSE) as a function of the iterations of the K-SVD algorithm. This graph corresponds to a gradual increment of atoms in the representation (L) from 1 to 7.

consists of two main processes: An offline K-SVD training process and an online image compression/decompression processes.

3.1.1. K-SVD training

The K-SVD training process is an off-line procedure, preceding any image compression. The training produces a set of K-SVD dictionaries that are then considered fixed for the image compression stage. A single dictionary is trained for each 15×15 patch over a set of examples that will referred to as the learning set. The training follows the description from the previous section, with parameters detailed in Section 4. Prior to the training process for each patch, the mean patch image of the examples in the learning set is calculated and subtracted from all the examples in this set.

The image compression process uses the following steps in the encoder, followed by a mirror application of them at the decoder:

3.1.2. Pre-processing

We use the same pre-process stage of geometrical warping as in [17]. This leads to better conditioning of the input image, increasing its redundancy versus the image database. The parameters of the warp are sent as side information for the inverse warp at the decoder. Those parameters are coded using 20 bytes. The pre-processing stage also includes a simple scale-down by a factor of 2:1 of the input image, so that later processing is applied on a smaller image.

3.1.3. Slicing to patches

Our approach also works on fixed size square patches, coding each patch separately and adaptively as in [17]. However, our coding strategy is different, being based on sparse and redundant representations. The change in methods leads to the ability to handle larger patches, and thus get more efficient coding. In our simulations we have used $N = 15$ pixels,[3] although larger patch sizes are also possible. The patches must also fit the dictionary in content, so the same mean patch images that were calculated for the learning set before the training process is subtracted out of the relevant patches.

[3] Due to the inconsistency between the image and patch sizes, some of the patches at the right and bottom boundaries are of different sizes than the rest.

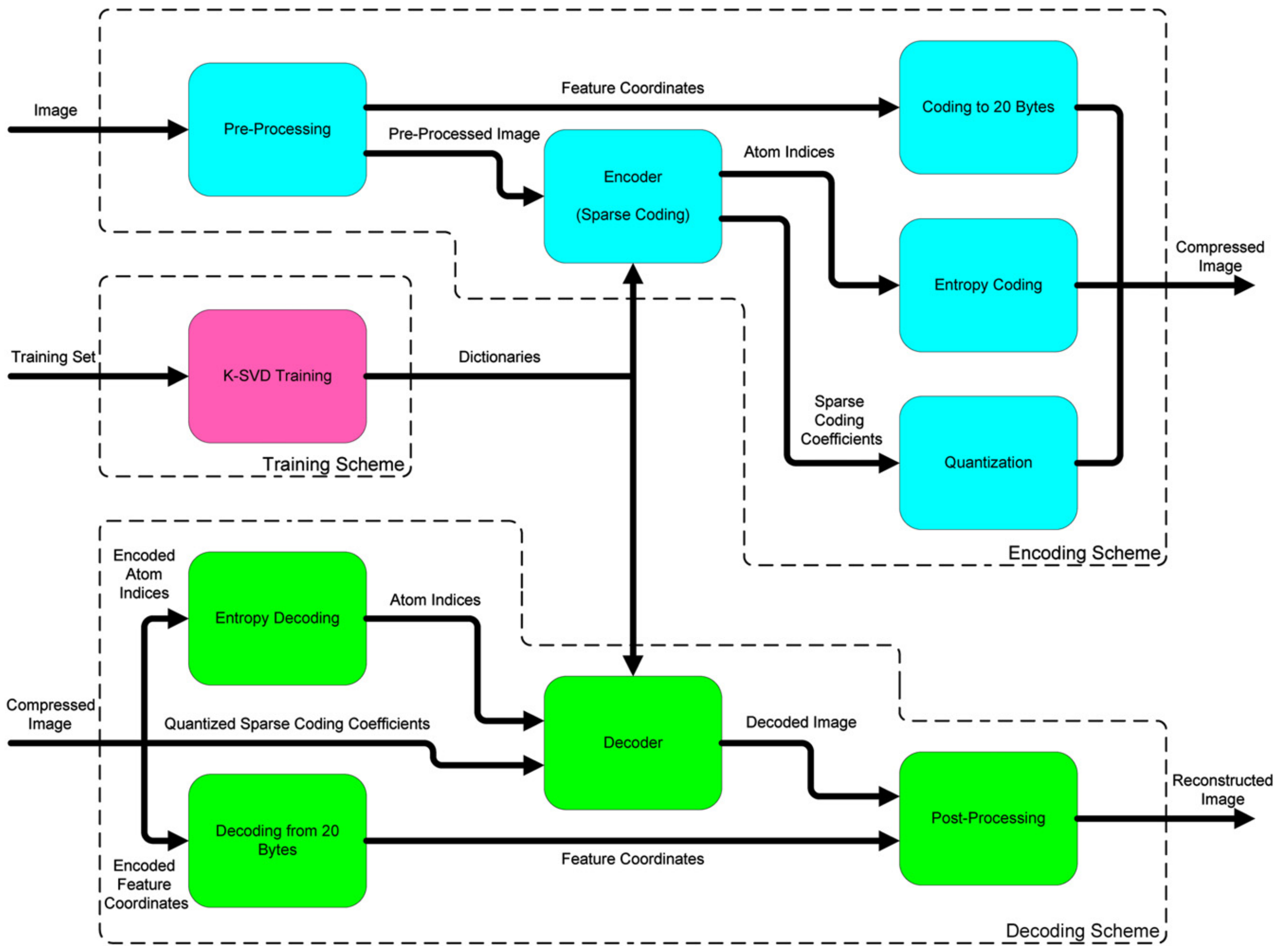

Fig. 3. Detailed block diagram of the proposed encoding/decoding method.

3.1.4. Sparse coding

Every patch location has a pre-trained dictionary of code-words (atoms) $\mathbf{D}_{ij}$. The size we use for k is 512—a discussion on this choice appears in Section 4. These dictionaries are generated using the K-SVD algorithm on a set of 4415 training examples. The coding itself is done by assigning a linear combination of few atoms to describe the patch content (sparse coding). Thus, information about the patch content includes both the linear weights and the involved indices. Note that the number of atoms vary from one patch to another, based on its average complexity, and this varied number is known at the decoder.

Both encoder and decoder are storing the dictionaries, just as in [17]. The sparse-coding stage for encoding of the patches is done using the OMP algorithm. The decoder is simply aggregating the proper atoms with the proper weights to reconstruct the patch, building the output one patch at a time independent of the others.

3.1.5. Entropy coding and quantization

We use a straight-forward Huffman table applied only to the indices. The representations' weights are quantized with a 7-bit uniform quantization, with boundaries chosen based on the training information for each patch separately.

Since the above compression scheme strongly relies on a successful detection of the features to warp by, a natural question is whether errors in this stage are destructive to the overall compression algorithm. When the detection of the features fails utterly, the compression necessarily leads to very low performance. Since we are performing a compression on an incoming image, we have access to both the resulting PSNR, and also the average PSNR we expect to get for this bits allocation. Thus, if the compression PSNR is below some threshold, we obviously know that the detection failed (or possibly the input image is not a face image). In such a case, and in a completely automatic fashion, we can employ the JPEG2000, which is indeed weaker, but still reasonable. A second option we have in such a system is to prompt the user to detect the features manually for this image. Thus, even rare situations where the features are not found do not lead to a breakdown in a system that employs the proposed algorithm.

3.2. VQ versus sparse representations

As mentioned in Section 2, the destination compression ratio and the number of training examples pose a hard limit on the allowed patch size, when using VQ. This is because as the patch size grows while keeping the rate (bits per pixel) fixed, the dictionary is required to grow exponentially, reaching sizes far beyond the number of examples to train on. More specifically, using patch size of $N \times N$ pixels, with k codewords in the VQ dictionary, considering an image with P pixels, and a target rate of B bits for the entire image, all these ingredients are related to each other by

$$B = \frac{P}{N^2} \cdot \log_2(k) \Rightarrow k = 2^{\frac{N^2 B}{P}}. \tag{3}$$

For example, coding an image of size 180×220 (P = 39,600) pixels[4] with a target rate of $B = 500$ bytes, a patch of size $N = 8$ leads to

[4] This is close to the actual size we use in our method, scaling down the original image as mentioned earlier by 2:1 in each axis as part of the pre-process stage. The actual size is 179×221.

$k = 88$, which is reasonable. However, increasing the patch size to $N = 12$ already leads to $k \approx 24{,}000$, which cannot be trained from 6000 examples. Thus, the VQ approach is limited to relatively small patch sizes (8×8 used in [17]), implying that much of the spatial redundancy between adjacent patches is overlooked.

When turning to sparse and redundant dictionaries, the picture changes dramatically. We consider a patch size of $N \times N$ pixels, with k atoms in the dictionary, L of which are used on average[5] in the representation (assuming 7-bit quantization of the weights). Handling of an image with P pixels, and a target rate of B bits for the entire image, all these ingredients lead to the relation

$$B = \frac{P}{N^2} \cdot L \cdot (\log_2(k) + 7). \tag{4}$$

This means that per patch, instead of using $\log_2 k$ bits when working with VQ, we need $L(\log_2 k + 7)$ bits on average. Thus, if the required amount of bits per patch is too high (as indeed happens when N grows), it can be absorbed by varying L. For example, for an image with $P = 39{,}600$ pixels, using $k = 512$ atoms, a target rate of $B = 500$—1000 bytes, and a patch sizes in the range $N = 8 \sim 30$, all these lead to a required average number of atoms L as shown in Fig. 4. As can be seen, the values are reasonable and so we have the flexibility to control the rate by changing L in the range 1—12.

Aside from the flexibility that the *Sparseland* model provides, this model also proposes a flexible description of the patches' content, in a way that leads to high coding performance. The fact that the dictionaries are trained from relevant examples leads to very effective and tight model. Indeed, this model has been deployed in recent years to a variety of applications in signal and image processing, leading typically to state-of-the-art results [29]. More work is required to explain the origin of this model's strength, though.

3.3. Run-time and memory requirements

In assessing complexities, we should separate the discussion between the training and the compression processes. The training process is composed of iterating on two main stages—sparse coding, which is done using the OMP algorithm, and dictionary update. As mentioned in [29], both stages can be done efficiently in $O(JnkL_{\mathrm{Tr}}S_{\mathrm{L}})$ per patch, where J is the number of iterations, n is the size of the patch, k is the number of atoms in the dictionary, L_{Tr} is the maximal number of non-zero elements in each coefficient vector in each patch during the training, and S_{L} is the number of examples in the learning set.

Since the training process is done to the entire learning set and to each patch separately, The complexity of training all the patches is $\mathrm{O}(\frac{P}{n}JnkL_{\mathrm{Tr}}S_{\mathrm{L}}) = \mathrm{O}(PJkL_{\mathrm{Tr}}S_{\mathrm{L}})$, where P is the number of pixels in the image and $\frac{P}{n}$ is therefore approximately (due to border issues) the number of disjoint square patches.

The compression process is also composed of two stages—the encoding and the decoding stages. The encoding is done using the OMP algorithm, which has a complexity of $\mathrm{O}(nkL_{\mathrm{Av}})$ per image, where L_{Av} is the average number of non-zero elements in the coefficient vector in all the patches. The encoding stage is done to each patch separately, and therefore the complexity of the entire encoding process is $\mathrm{O}(\frac{P}{n}nkL_{\mathrm{Av}}) = \mathrm{O}(PkL_{\mathrm{Av}})$. The decoding stage is simply calculating the linear combination result, which can be done in $\mathrm{O}(nL_{\mathrm{Average}})$ per patch. The overall complexity of this stage is therefore $\mathrm{O}(\frac{P}{n}nL_{\mathrm{Av}}) = \mathrm{O}(PL_{\mathrm{Av}})$.

[5] If L varies from one patch to another, side information is necessary in order to instruct the decoder how many atoms to use in each patch. However, in our scheme, while L indeed varies spatially, it remains fixed for all images and thus the decoder knows the atom allocated per patch with no additional side information.

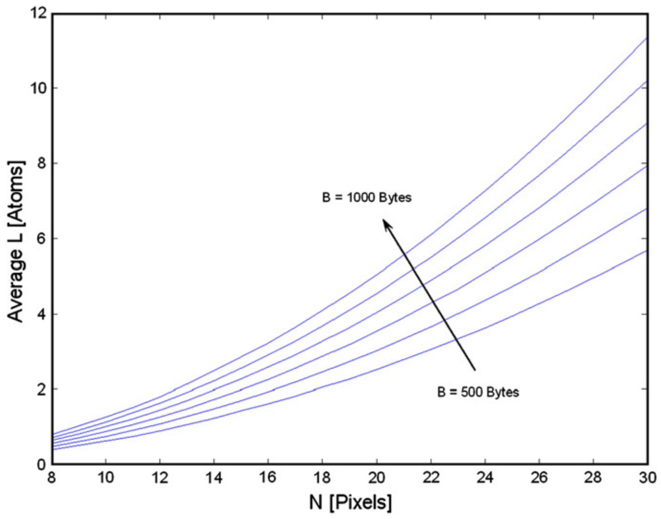

Fig. 4. The Required average number of atoms L as a function of the patch size N, with the influence of the overall allocated number of bits B.

In assessing the required memory for the proposed compression method we need to take into account the dictionaries that were produced for each patch, the previously mentioned mean patch images, the Huffman tables, the coefficient usage tables and the quantization levels for each patch. Calculating in bytes, the required memory for the above mentioned data is given roughly by $\mathrm{O}(Pk)$ bytes. In our tests with $P = 39{,}600$ and $k = 512$ this leads to less than 20 Mbytes.

Practically, the training and testing processes has been all implemented using non-optimized Matlab code on a regular PC (Pentium 2, 2 GHz, 1 GByte RAM). Training of all the required dictionaries requires 100 h to complete (and thus has been implemented on several computers in parallel, each handling a different group of patches). The compression of a single image requires 5 s, and its decompression takes less than 1 s.

4. More details and results

We conducted several experiments in order to evaluate the performance of our compression method and the quality of its resulting images. In this section we show some statistical results as well as reconstructed images from our compression method and a comparison between our results to several other competitive compression techniques.

We used 4415 face images as our learning set and a different 100 images as our test set. All the images in both sets are of a fixed size of 358×441 pixels prior to the pre-processing, and of size 179×221 after a simple scale-down. The slicing to patches is uniform over the image with the previously mentioned size, and exceptions at the borders. Examples of such pre-processed training/testing images can be seen in Fig. 5.

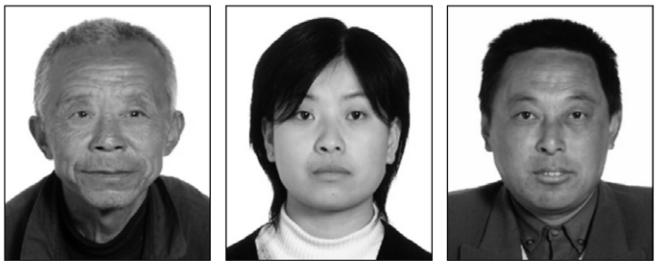

Fig. 5. Examples of pre-processed images used in the training/testing sets.

Before turning to preset the results we should add the following: while all the results shown here refer to the specific database we operate on, the overall scheme proposed is general and should apply to other face images databases just as well. Naturally, some changes in the parameters might be necessary, and among those, the patch size is the most important to consider. We also note that as one shifts from one source of images to another, the relative size of the background in the photos may vary, and this necessarily leads to changes in performance. More specifically, when the background regions are larger (e.g., the images we use here have relatively small such regions), the compression performance is expected to improve.

4.1. K-SVD dictionaries

The primary stopping condition for the training process was set to be a limitation on the maximal number of K-SVD iterations (being 100). A secondary stopping condition was a limitation on the minimal representation error. In the image compression stage we added a limitation on the maximal number of atoms per patch. These conditions were used to allow us to better control the rates of the resulting images and the overall simulation time.

Every obtained dictionary contains 512 patches of size 15×15 pixels as atoms. In Fig. 6 we can see the dictionary that was trained for patch number 80 (The left eye) for $L = 4$ sparse coding atoms, and similarly, in Fig. 7 we can see the dictionary that was trained for patch number 87 (The right nostril) also for $L = 4$ sparse coding atoms. It can be seen that both dictionaries contain images similar in nature to the image patch for which they were trained for. A similar behavior was observed in other dictionaries.

4.2. Reconstructed images

Our coding strategy allows us to learn which parts of the image are more difficult than others to code. This is done by assigning the same representation error threshold to all of the patches, and observing how many atoms are required for the representation of each patch on average. Clearly, patches with a small number of allocated atoms are simpler to represent than others. We would expect that the representation of smooth areas of the image such as the background, parts of the face and maybe parts of the clothes will be simpler than the representation of areas containing high frequency elements such as the hair or the eyes. Fig. 8 shows maps of atom allocation per patch and representation error (RMSE—squared-root of the mean squared error) per patch for the images in the test set in two different bit-rates. It can be seen that more atoms were allocated to patches containing the facial details (hair, mouth, eyes, and

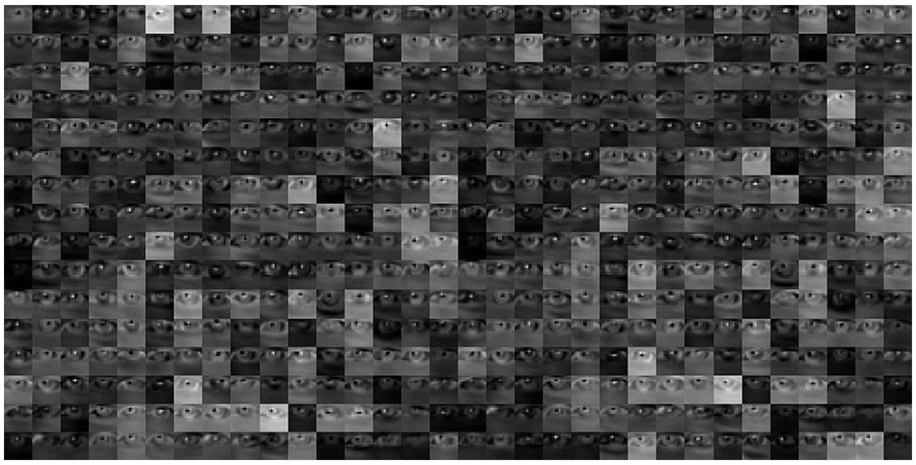
Fig. 6. The Dictionary obtained by K-SVD for Patch No. 80 (the left eye) using the OMP method with $L = 4$.

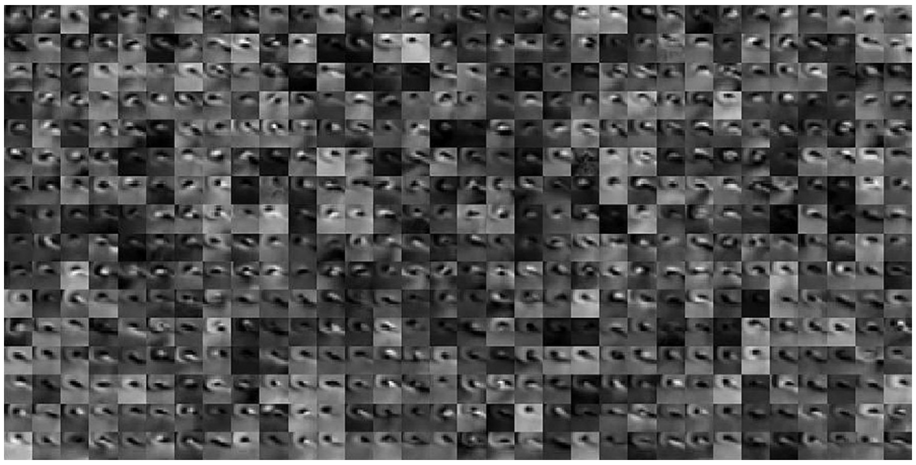
Fig. 7. The Dictionary obtained by K-SVD for Patch No. 87 (the right nostril) using the OMP method with $L = 4$.

facial borders), and that the representation error is higher in these patches. It can also be seen that the overall number of allocated atoms increases and the representation error decreases with the image bit-rate.

As in other compression methods, our reconstructed images suffer from several kinds of artifacts. These artifacts are caused by the slicing to disjoint patches, coding them independently. Contrary to methods such as JPEG and JPEG2000, the reconstructed images in our method do not have a strong smearing artifact all over the image, but only local small areas in which the image is smeared. Other artifacts include blockiness effect due to the slicing to patches, inaccurate representation of high frequency elements in the image, inability to represent complicated textures (mainly in the clothes) and inconsistency in the spatial location of several facial elements comparing to the original image. These artifacts are caused by the nature of our compression method, which has a limited ability to build the reconstructed image out of the given dictionaries. In Fig. 9 we can see two original images from the learning set and their reconstructed images in a bit-rate of 630 bytes. The mentioned artifacts can be seen in these images especially in the areas of the chin, the neck, the mouth, the clothes and the outline of the hair. Although there are areas in the images in which there is a smearing effect, the majority of the image is clear and sharp, and certainly in a high visual quality.

Fig. 10 shows three original images from the test set and their reconstructed images in a bit-rate of 630 bytes. As in the previous images, the mentioned artifacts can be seen in these images as well, in the same image areas as before. As expected, the quality of the images from the test set is not as high as the quality of images from the learning set, but these images too are clear and sharp, and in a high visual quality.

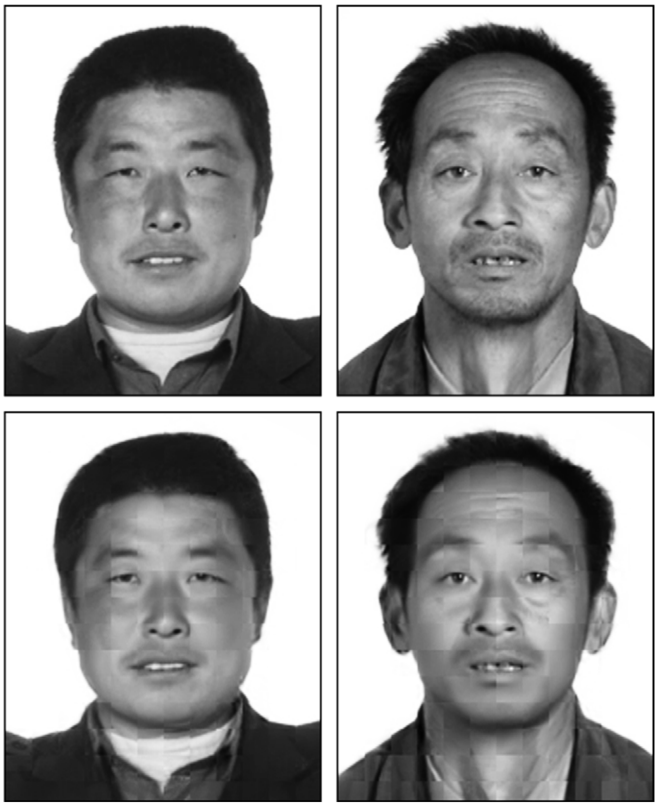

Fig. 9. Examples of original (top) and reconstructed (bottom) images from the learning set, with a "good" 4.04 (on the left) and a "bad" 5.9 (on the right) representation RMSE, both using 630 bytes.

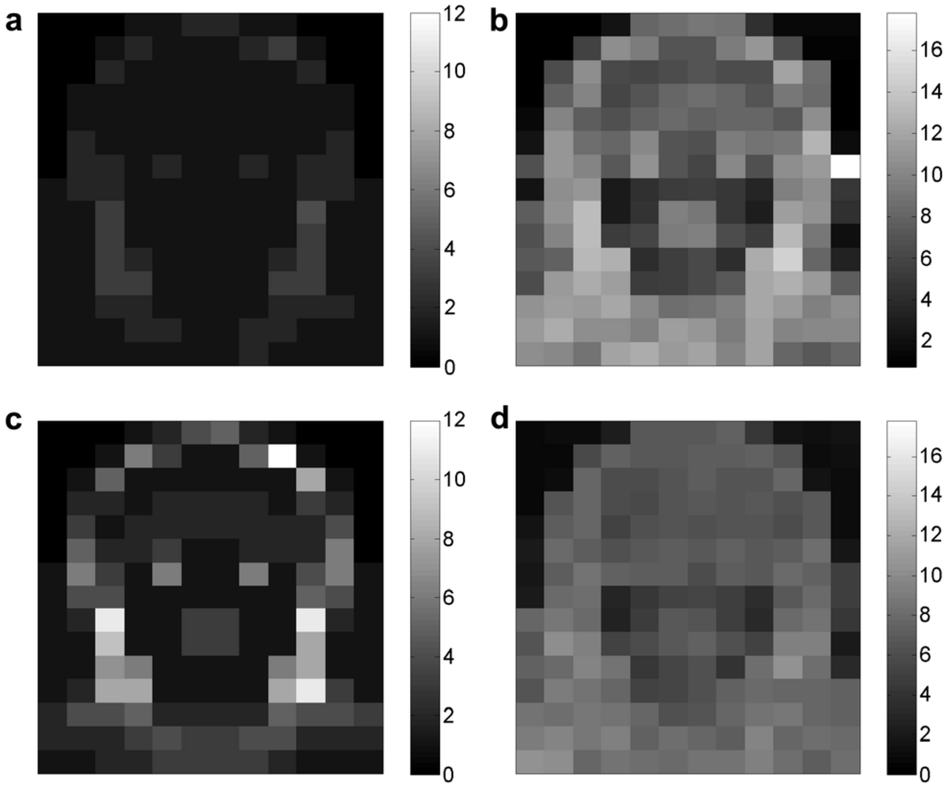

Fig. 8. (a) The atom allocation map for 400 bytes, (b) the representation error map for 400 bytes, (c) the atom allocation map for 820 bytes, and (d) the representation error map for 820 bytes.

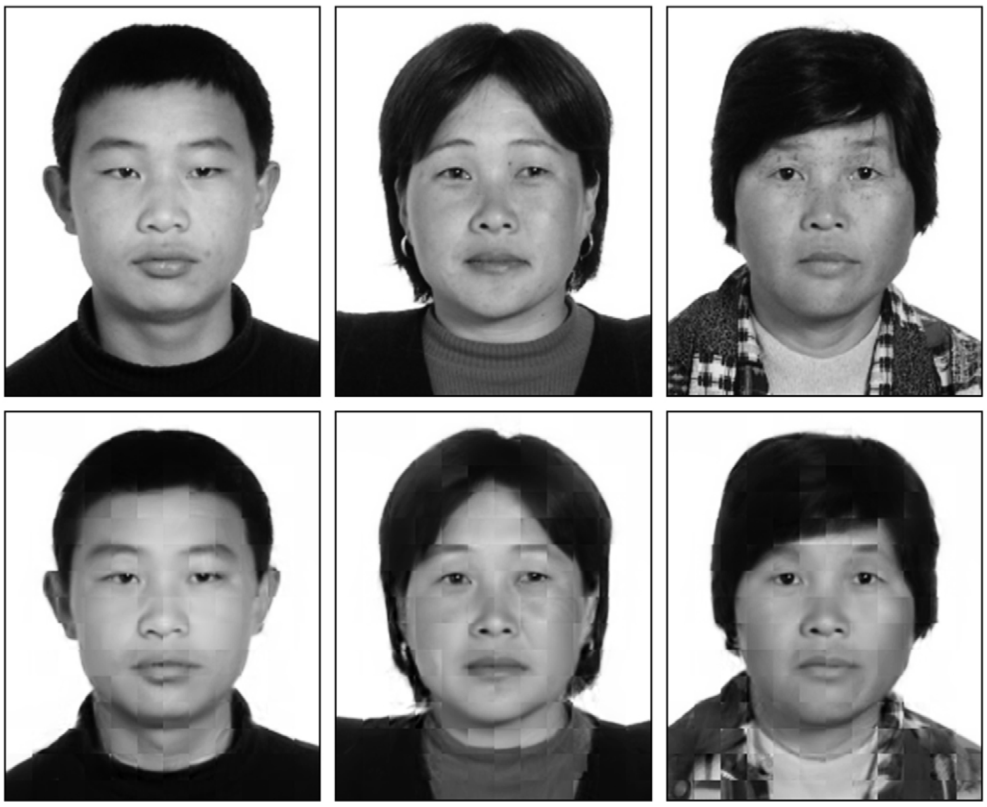

Fig. 10. Examples of original and reconstructed images from the test set, with a "good" 5.3 (left), an "average" 6.84 (middle) and a "bad" 11.2 (right) representation RMSE, all three using 630 bytes.

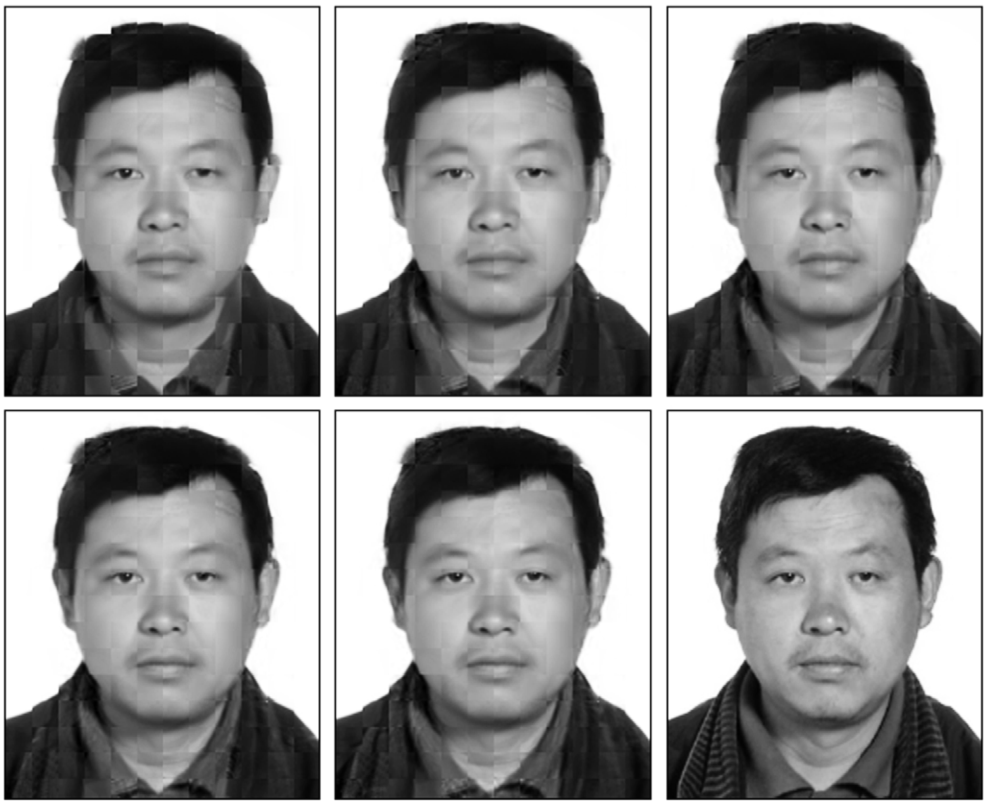

Fig. 11. Comparing the visual quality of a reconstructed image from the test set in several bit-rates. From top left,clockwise: 285 bytes (RMSE 10.6), 459 bytes (RMSE 8.27), 630 bytes (RMSE 7.61), 820 bytes (RMSE 7.11), 1021 bytes (RMSE 6.8), original image.

Much like other compression methods, the quality of the reconstructed images in our method improves as the bit-rate increases. However, the contribution gained from such a rate increment is not divided equally over the image. Additional bits are allocated to patches with higher representation error, and those are improved first. This property is directly caused by the nature of the compression process, which is RMSE oriented and not bit-rate oriented. The compression process sets a single RMSE threshold for all the patches, forcing each of them to reach it without fixing the number of allocated atoms per patch. Patches with simple (smooth) content are most likely to have a representation error far below the threshold even using zero or one atom, whereas patches with more complex content are expected to give a representation error very close to the threshold. Such problematic patches will be forced to improve their representation error by increasing the number of atoms they use as the RMSE threshold is decreased, while patches with a representation error below the threshold will not be forced to change at all. Fig. 11 illustrates the gradual improvement in the image quality as the bit-rate increases. As can be seen, not all the patches improve as the bit-rate increases but only some of them, such as several patches in the clothes area, in the ears and in the outline of the hair. These patches were more difficult to represent than others.

4.3. Comparing to other techniques

An important part in assessing the performance of our compression method is its comparison to known and competitive compression techniques. As mentioned before, we compare our results in this work with JPEG, JPEG2000, The VQ-Based compression method described in [17], and a PCA-Based compression method that was built especially for this work as a competitive benchmark. We therefore start with a brief description of the PCA technique.

The PCA-Based compression method is very similar to the scheme described in this work, simply replacing the K-SVD dictionaries with a Principal Component Analysis (PCA) ones. These dictionaries are square matrices storing the eigenvectors of the autocorrelation matrices of the training examples in each patch, sorted by a decreasing order of their corresponding eigenvalues.

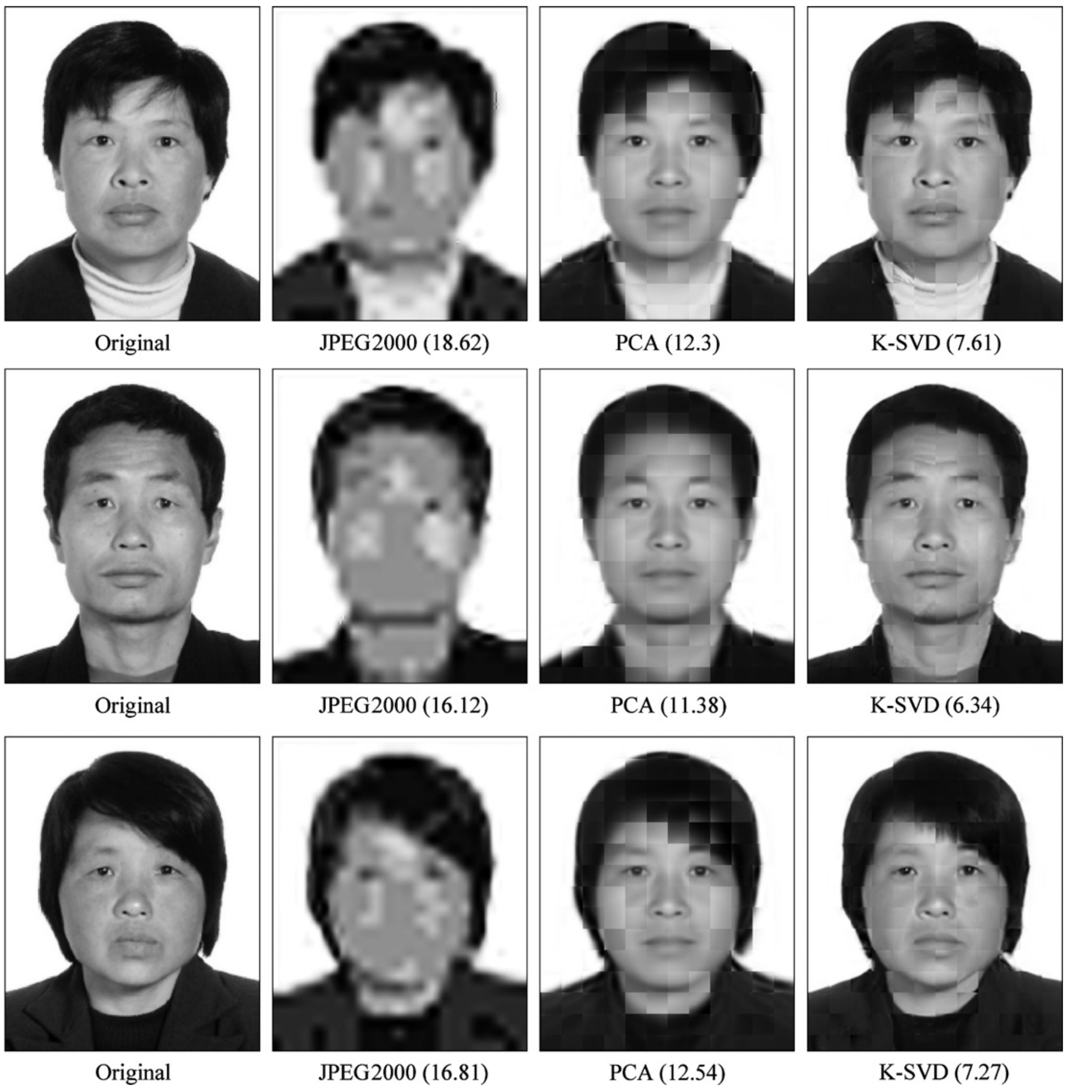

Fig. 12. Facial images compression with a bit-rate of 400 bytes. Comparing results of JPEG2000, the PCA results, and our K-SVD method. The values in the brackets are the representation RMSE.

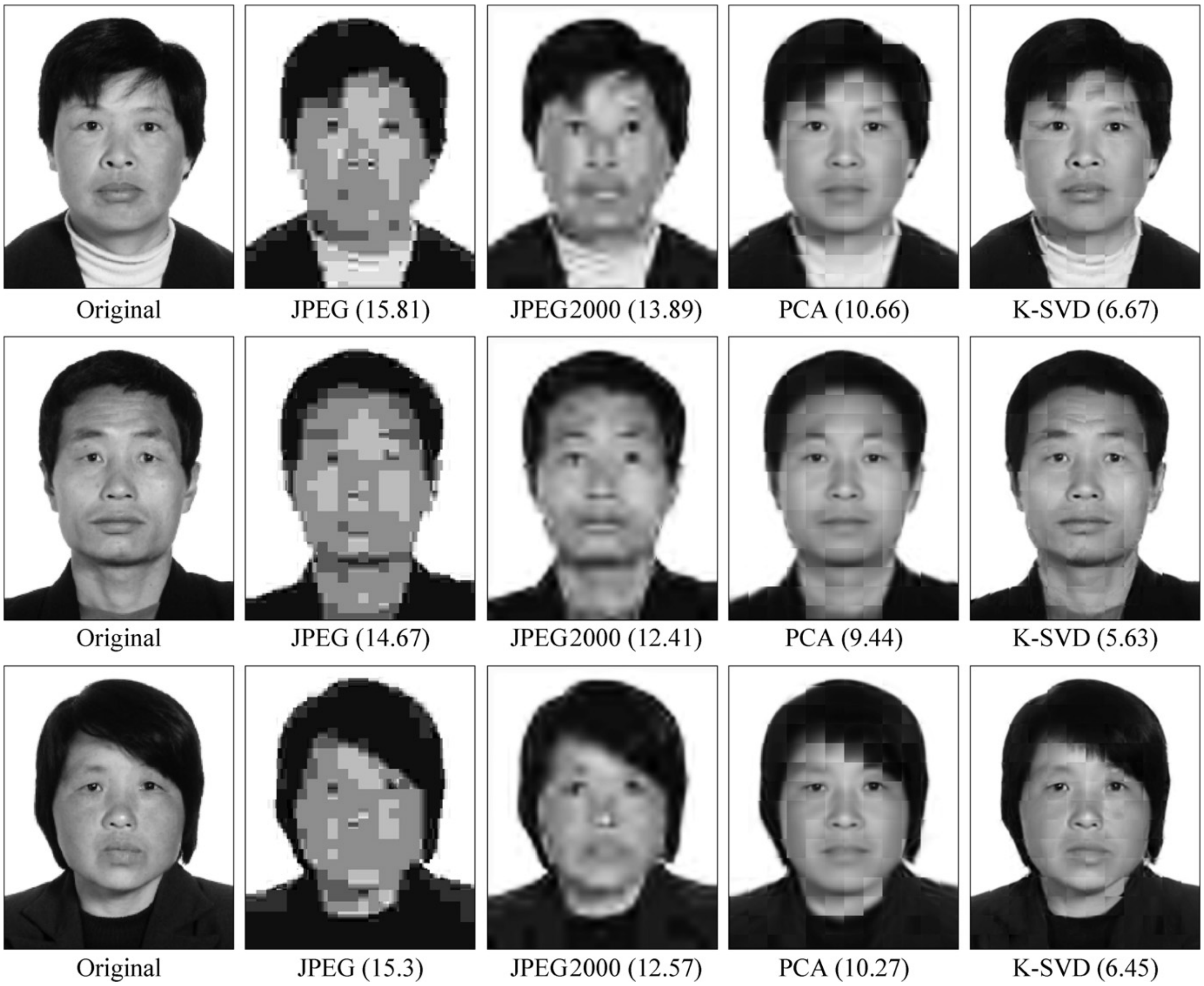

Fig. 13. Facial images compression with a bit-rate of 550 bytes. Comparing results of JPEG, JPEG2000, the PCA results, and our K-SVD method. The values in the brackets show the representation RMSE.

Another induced difference is the replacement of the sparse coding representation with a simple linear approximation that leans on the first several eigenvectors, till the representation error decrease below the specified threshold. Comparison to the PCA-Based compression method is important in this work, because whereas our technique could be interpreted as an adaptation of the JPEG2000 to the image family by training, PCA could be seen as a similar step emerging from JPEG.

Figs. 12–14 show a visual comparison of our results with the JPEG, JPEG2000, and the PCA for three different bit-rates. Note that Fig. 12 does not contain JPEG examples because the rate is below the minimum possible one in this case. These figures clearly show the visual and quantitative advantage of our method over all the alternatives. We can especially notice the strong smearing effect in the JPEG and JPEG2000 images, whereas our images are clear and sharp.

Fig. 15 shows a Rate-Distortion curves comparison of the compression methods mentioned before, averaged on the 100 test images. 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.

In addition to these Rate-Distortion curves we added a curve for a "Clean" JPEG2000 method, which is simply a horizontally shifted version of the JPEG2000 curve taking into account the header embedded in the JPEG2000 standard. This header was found to be of size 220 bytes.

It can be seen from this comparative graph that the K-SVD compression method has the best performance and especially so in the very low bit-rates of 300–1000 bytes, which is our area of interest.

4.4. Dictionary redundancy

Dictionary redundancy, or overcompleteness, is defined as k/n, being 1 for complete (non-redundant) representations, and above this in our experiments. A natural question is whether such redundancy is truly necessary for getting the compression results shown. Fixing the patch size, n, what would be the effect of changing the parameter k? Fig. 16 shows four curves of the averaged PSNR on the test set images with varying

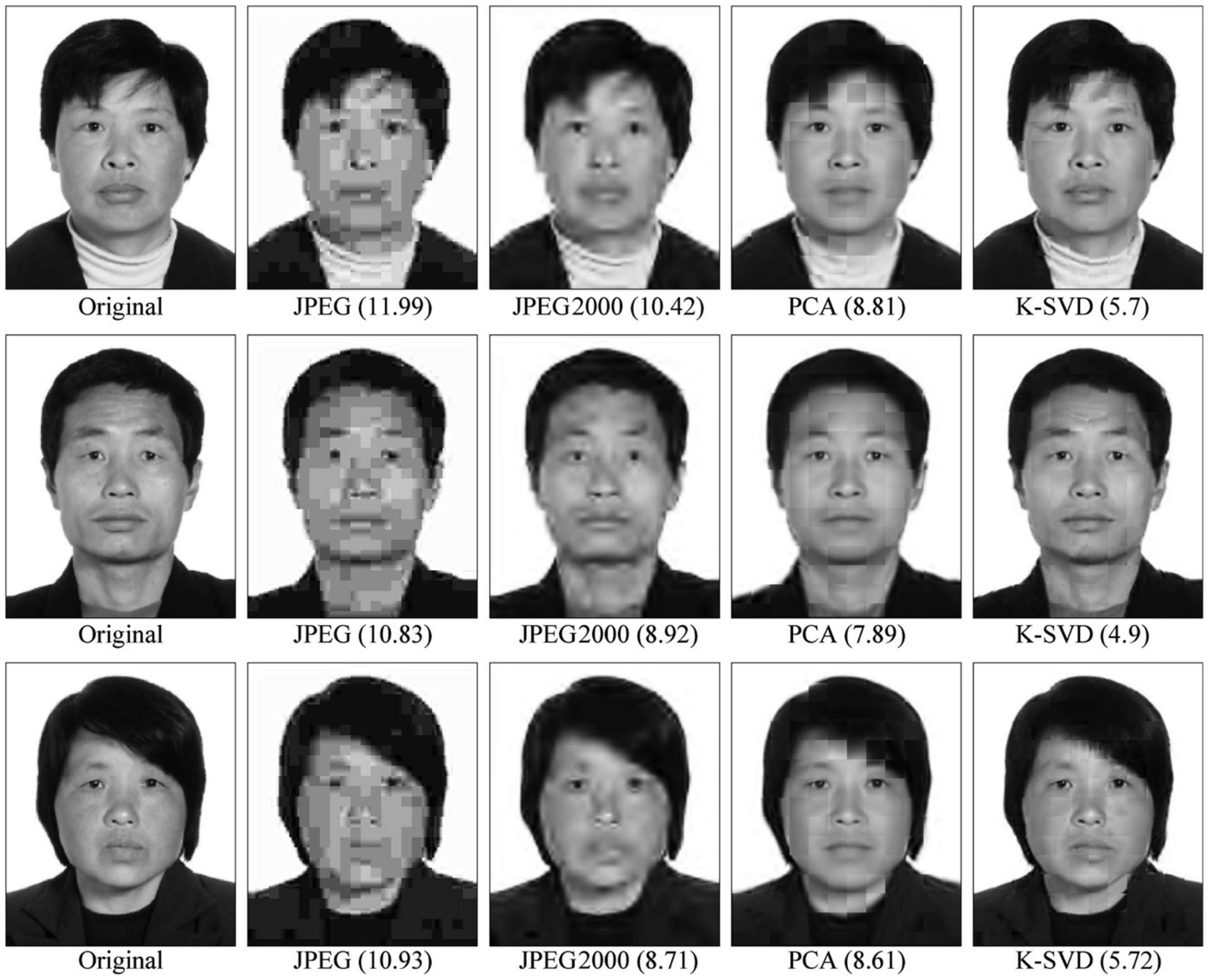

Fig. 14. Facial images compression with a bit-rate of 820 bytes. Comparing results of JPEG, JPEG2000, the PCA results, and our K-SVD method. The values in the brackets show the representation RMSE.

redundancy. Each curve shows this behavior for a different fixed image bit-rate. Note that since the patch size is 15×15 pixels, having 225 atoms implies a complete (non-redundant) dictionary, and with 150 atoms we are in fact in an undercomplete regime.

We can see in Fig. 16 that the quality of the images is improving (PSNR-wise) with the dictionary redundancy, but this increase is subsiding. Intuitively, we would expect a change in the orientation of the curves at some point, partly due to overfitting of the dictionaries (due to the limited number of examples), and partly because too high redundancy is expected to cause deterioration in performance. Such change in tendency has not been observed because of limitations in the training data size.

5. Conclusions

In this paper we present a facial image compression method, based on sparse and redundant representations and the K-SVD dictionary learning algorithm. The proposed compression method is tested in various bit-rates and options, and compared to several known compression techniques with great success. Results on the importance of redundancy in the deployed dictionaries are presented. The contribution of this work has several facets: first, while sparse and redundant representations and learned dictionaries have shown to be effective in various image processing problems, their role in compression has been less explored, and this work provides the first evidence to its success in this arena as well. Second, the proposed scheme is very practical, and could be the foundation for systems that use large databases of face images. Third, among the various ways to imitate the VQ and yet be practical, the proposed method stands as an interesting option that should be further explored.

As for future work, we are currently exploring several extensions of this activity, such as reducing or eliminating the much troubling blockiness effects due to the slicing to patches, generalization to compression of color images, and adopting the ideas in this work for compression of finger-prints images. The horizon and the ultimate goal, in this respect, is a successful harnessing of the presented methodology for general images, in a way that surpasses the JPEG2000 performance—we believe that this is achievable.

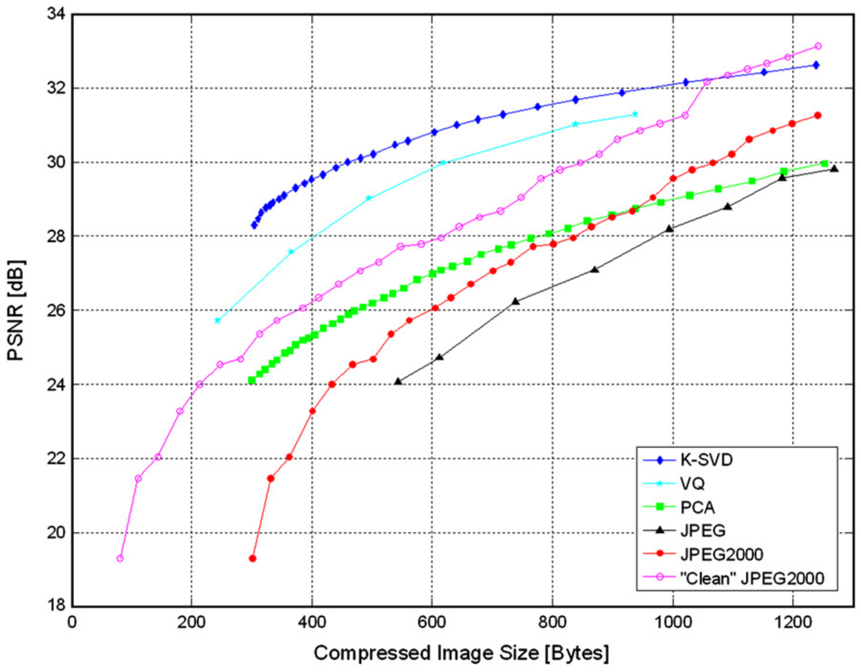

Fig. 15. Rate-Distortion curves for the JPEG, JPEG2000, "Clean" JPEG2000 (i.e., after removal of the header bits), PCA, VQ and the K-SVD methods.

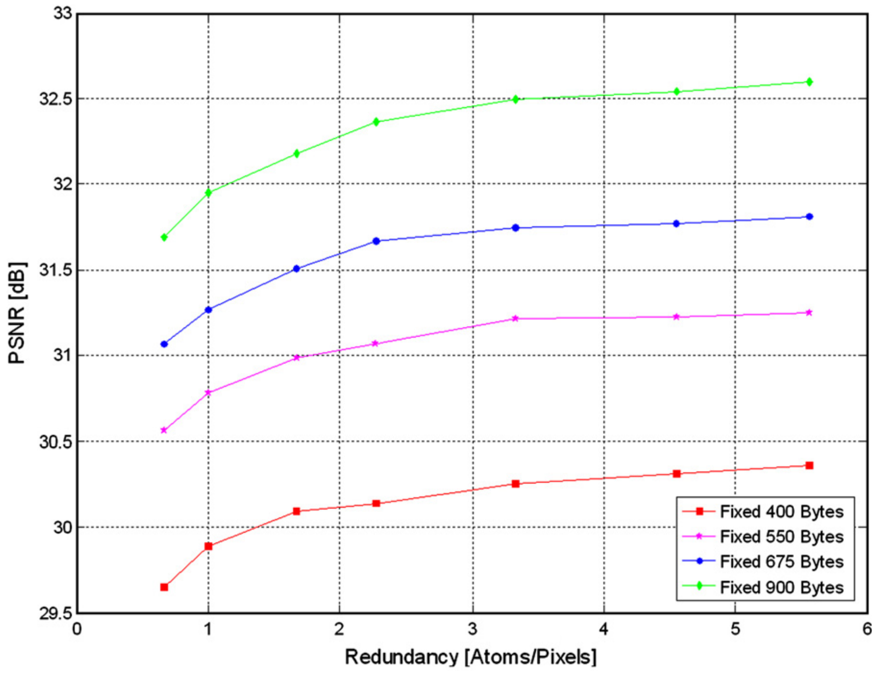

Fig. 16. The effect of redundancy in the trained dictionaries on the quality of the test set images in PSNR for a fixed bit-rate (four different fixed values).

Acknowledgments

The authors are grateful to Dr. Michal Aharon for her support and help throughout this work. We also thank Dr. Roman Goldenberg for sharing his work with us for comparison purposes.

References

[1] D.S. Taubman, M.W. Marcellin, JPEG2000: Image Compression Fundamentals, Standards and Practice, Kluwer Academic Publishers., Norwell, MA, USA, 2001.
[2] B. Moghaddam, A. Pentland, An automatic system for model-based coding of faces, in: Proceedings of DCC '95 Data Compression Conference, Snowbird, UT, USA, March 1995, pp. 28–30.
[3] J.H. Hu, R.S. Wang, Y. Wang, Compression of personal identification pictures using vector quantization with facial feature correction, Optical-Engineering 35 (1996) 198–203.
[4] A. Lanitis, C.J. Taylor, T.F. Cootes, Automatic interpretation and coding of face images using flexible models, IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (1997) 743–756.
[5] M. Sakalli, H. Yan, K.M. Lam, T. Kondo, Model-based multi-stage compression of human face images, in: Proceedings of 14th International Conference on Pattern Recognition (ICPR), Brisbane, Qld., Australia. August 1998, pp. 16–20.
[6] M. Sakalli, H. Yan, Feature-based compression of human face images, Optical-Engineering 37 (1998) 1520–1529.

[7] J. Huang, Y. Wang, Compression of color facial images using feature correction two-stage vector quantization, IEEE Transactions on Image Processing 8 (1999) 102–109.
[8] M. Sakalli, H. Yan, A. Fu, A region-based scheme using RKLT and predictive classified vector quantization, Computer Vision and Image Understanding 75 (1999) 269–280.
[9] A. Shashua, A. Levin, Linear image coding for regression and classification using the tensor-rank principle, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Kauai, HI, USA, December 2001, pp. 8–14.
[10] A.J. Ferreira, M.A.T. Figueiredo, Class-adapted image compression using independent component analysis, in: Proceedings of the International Conference on Image Processing (ICIP), Barcelona, Spain, September 2003, pp. 14–17.
[11] A.J. Ferreira, M.A.T. Figueiredo, Image compression using orthogonalized independent components bases, in: Proceedings of the IEEE XIII Workshop on Neural Networks for Signal Processing, Toulouse, France, September 2003, pp. 17–19.
[12] O.N. Gerek, C. Hatice, Segmentation based coding of human face images for retrieval, Signal-Processing 84 (2004) 1041–1047.
[13] K. Inoue, K. Urahama, DSVD: a tensor-based image compression and recognition method, in: Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Kobe, Japan, Mat 2005, pp. 23–26.
[14] Z. Qiuyu, W. Suozhong, Color personal ID photo compression based on object segmentation, in: Proceedings of the Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), Victoria, BC, Canada, August 2005, pp. 24–26.
[15] T. Hazan, S. Polak, A. Shashua, Sparse image coding using a 3D non-negative tensor factorization, in: Proceedings of the 10th IEEE International Conference on Computer Vision (ICCV), Beijing, China, October 2005, pp. 17–21.
[16] A.J. Ferreira, M.A.T. Figueiredo, On the use of independent component analysis for image compression, Signal Processing: Image Communication 21 (2006) 378–389.
[17] M. Elad, R. Goldenberg, R. Kimmel, Low bit-rate compression of facial images, IEEE Transactions on Image Processing 16 (2007) 2379–2383.
[18] S. Mallat, Z. Zhang, Matching pursuits with time-frequency dictionaries, IEEE Transactions on Signal Processing 41 (1993) 3397–3415.
[19] Y.C. Pati, R. Rezaiifar, P.S. Krishnaprasad, Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition, in: Conf. Rec. 27th Asilomar Conf. Signals, Syst. Comput., 1993.
[20] J.A. Tropp, Greed is good: algorithmic results for sparse approximation, IEEE Transactions on Information Theory 50 (2004) 2231–2242.
[21] J.A. Tropp, Just relax: convex programming methods for subset selection and sparse approximation, IEEE Transactions on Information Theory 52 (2004) 1030–1051.
[22] S.S. Chen, D.L. Donoho, M.A. Saunders, Atomic decomposition by basis pursuit, SIAM Review 43 (2001) 129–159.
[23] I.F. Gorodnitsky, B.D. Rao, Sparse signal reconstruction from limited data using FOCUSS: a re-weighted norm minimization algorithm, IEEE Transactions on Signal Processing 45 (1997) 600–616.
[24] D.L. Donoho, X. Huo, Uncertainty principles and ideal atomic decomposition, IEEE Transactions on Information Theory 47 (1999) 2845–2862.
[25] D.L. Donoho, M. Elad, Optimally sparse representation in general (non-orthogonal) dictionaries via ℓ^1 minimization, Proceedings of the National Academy of Sciences of the United States of America 100 (2003) 2197–2202.
[26] D.L. Donoho, M. Elad, V. Temlyakov, Stable recovery of sparse overcomplete representations in the presence of noise, IEEE Transactions on Information Theory 147 (2007) 185–195.
[27] M. Aharon, M. Elad, A.M. Bruckstein, On the uniqueness of overcomplete dictionaries, and a practical way to retrieve them, Journal of Linear Algebra and Applications 416 (2006) 48–67.
[28] M. Aharon, M. Elad, A.M. Bruckstein, The K-SVD: an algorithm for designing of overcomplete dictionaries for sparse representation, IEEE Transactions on Signal Processing 54 (2006) 4311–4322.
[29] M. Elad, M. Aharon, Image denoising via sparse and redundant representations over learned dictionaries, IEEE Transactions on Image Processing 15 (2006) 3736–3745.
[30] Y. Linde, A. Buzo, R.M. Gray, An algorithm for vector quantiser design, IEEE Transactions on Communications 28 (1980) 84–95.
[31] A. Gersho, R.M. Gray, Vector Quantization And Signal Compression, Kluwer Academic Publishers, Dordrecht, Netherlands, 1992.
[32] P. Cosman, R. Gray, M. Vetterli, Vector quantization of images subbands: a survey, IEEE Transactions on Image Processing 5 (1996) 202–225.
[33] P.J. Burt, E.H. Adelson, The Laplacian pyramid as a compact image code, IEEE Transactions on Communications 31 (1983) 532–540.
[34] L. Peotta, L. Granai, P. Vandergheynst, Image compression using an edge adapted redundant dictionary and wavelets, Signal-Processing 86 (2006) 444–456.