ELSEVIER

Signal Processing 82 (2002) 1559–1580

SIGNAL PROCESSING

www.elsevier.com/locate/sigpro

Multidimensional signal compression using multiscale recurrent patterns

Murilo B. de Carvalho[a], Eduardo A.B. da Silva[b, *], Weiler Alves Finamore[c]

[a] *TET/CTC, Universidade Federal Fluminense, R. Passos da Pátria, 156 Niteroi, RJ 24210-240, Brazil*
[b] *PEE/COPPE/DEL/EE, Universidade Federal do Rio de Janeiro, Cx. P. 68504, Rio de Janeiro, RJ 21945-970, Brazil*
[c] *CETUC, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ, Brazil*

Received 1 April 2001

Abstract

In this paper we propose a new multidimensional signal lossy compression method based on multiscale recurrent patterns, referred to as multidimensional multiscale parser (MMP). In it, a multidimensional signal is recursively segmented into variable-length vectors, and each segment is encoded using expansions and contractions of vectors in a dictionary. The dictionary is updated while the data is being encoded, using concatenations of expanded and contracted versions of previously encoded vectors. The only data encoded are the segmentation tree and the indexes of the vectors in the dictionary, and therefore no side information is necessary for the dictionary updating. The signal segmentation is carried out through a rate–distortion optimization procedure. A two-dimensional version of the MMP algorithm was implemented and tested with several kinds of image data. We have observed that the proposed dictionary updating procedure is effective in adapting the algorithm to a large variety of image content, lending to it a universal flavor. For text and graphics images, it outperforms the state-of-the-art SPIHT algorithm by more that 3 dB at 0.5 bpp, while for mixed document images, containing text, graphics and gray-scale images, by more than 1.5 dB at the same rate. Due to the way the images are segmented, they appear slightly blocky at low rates. We have alleviated this problem by proposing an effective way of reducing the blockiness in the reconstructed image, with no penalty in signal-to-noise ratio performance in most cases. We conclude the paper with a theoretical analysis of the approximate matching of Gaussian vectors using scales, which gives a justification of why approximate multiscale matching is a good option, specially at low rates.

Keywords: Recurrent pattern matching; Multiscale decomposition; Multidimensional signal compression; Vector quantization

1. Introduction

Most methods representing the state-of-the-art in image and video compression schemes attempt to encode a source using the transformation–quantization–entropy coding paradigm. For example, in the JPEG [19] and MPEG [17] standards, the image or motion-compensated frame difference is first transformed into a set of coefficients using the block DCT. The generated coefficients are then scalar quantized, and the resulting symbols are encoded using run-length encoding followed by Huffman coding. In the EBCOT image compression scheme, employed in the recently developed JPEG2000 standard [14], the

* Corresponding author.

E-mail addresses: murilo@lps.ufrj.br (M.B. de Carvalho), eduardo@lps.ufrj.br (E.A.B. da Silva), weiler@cetuc.puc-rio.br (W.A. Finamore).

transform used is the wavelet transform, the coefficients are quantized using successive approximation scalar quantization and the generated symbols are encoded using arithmetic encoding. The success of such schemes relies on the supposition that the image data is essentially of a lowpass nature, so that most of their important information will be clustered in the low-frequency transform coefficients, leading to efficient encoding schemes for the quantized coefficients. However, this assumption is not true for a large class of image data, as for example, text and graphics. Such images must be dealt with using dedicated algorithms. A problem arises when the images are mixed, that is, contain text and graphics. The usual approach is to adaptively switch from encoding methods depending on the type of image content [5]. Another case in which transform-based methods do not provide a definitive answer is in the encoding of motion-compensated frame differences. For such signals the energy compaction property of transforms is in general not valid.

Taking the above into consideration, if one wants to overcome the intrinsic limitations of transform-based methods, it is worth pursuing alternative signal compression methods that do not rely on the transformation–quantization–encoding paradigm. Such methods should be able to automatically adapt to a large variety of image content. Also, they should perform well for a wide range of bitrates. Another desirable characteristic is that they should be easily adaptable to multidimensional signals.

In this paper, we propose a multidimensional signal compression method that we refer to as multidimensional multiscale parser (MMP). In it, a signal is encoded by first segmenting it into variable-length blocks. Each block is encoded using expansions and contractions of vectors from a dictionary. This is where the term “multiscale” is derived from. The dictionary is updated as the data is being encoded, with concatenations of expansions and contractions of previously encoded vectors. We can view MMP as an encoder using multidimensional multiscale recurrent patterns. It usually begins with a very simple dictionary, and learns its vectors from the data itself. Therefore, the MMP algorithm tends to work well for a variety of image data, and also adapts well for non-stationary sources. It works well from low to high rates, having also lossless capability. Since its signal segmentation procedure can be easily described in multiple dimensions, its multidimensional extensions are trivial.

This paper is organized as follows. In Section 2 the problem of approximate matching with scales is described, and a distortion-controlled, one-dimensional version of the MMP algorithm is presented. In Section 3 we propose a rate–distortion optimized version of MMP. Section 4 presents the extension of the MMP algorithm to multidimensional sources, while Section 5 proposes a fast MMP implementation using multiple codebooks. Experimental results with image data are presented in Section 6. Section 7 describes a method for the reduction of blocking effects in MMP and Section 8 presents the conclusions. Appendix A contains a mathematical analysis of the matching of Gaussian vectors using multiple-scale dictionaries, that serves as a justification for the use of scales in MMP.

2. The MMP algorithm

In this work, we study methods to lossy compress data using a new concept that we call *approximate matching with scales*. The approximate matching with scales is an extension of ordinary approximate pattern matching, where we allow vectors of different lengths to be matched. In order to do this, we use a scale transformation $T_N^M : \mathbb{R}^{\mathbb{M}} \mapsto \mathbb{R}^{\mathbf{N}}$ to adjust the sizes of the vectors prior to the matching attempt [7]. For example, if we want to use a vector $\mathbf{S}$ of length N' to represent another vector $\mathbf{X}$ of a different length N, we first compute $\mathbf{S}^s = T_N^{N'}[\mathbf{S}]$, a scaled version of $\mathbf{S}$ that has the same length N of $\mathbf{X}$ (we can also just write $\mathbf{S}^s = T_N[\mathbf{S}]$ meaning that $N' = \ell(\mathbf{S})$, that is, the length of $\mathbf{S}$). This operation is illustrated in Fig. 1.

We can then use $\mathbf{S}^s$ to replace $\mathbf{X}$ provided that they are close enough. This is illustrated in Fig. 2.

In Appendix A we perform a mathematical analysis of the matching properties of Gaussian vectors. There, we compare the performance of a VQ whose dictionary is built with blocks from the input signal, to the performance of another VQ, whose dictionary is built using scaled versions of input blocks. We conclude that, even for Gaussian memoryless sources, it can be advantageous to use a dictionary with scales, specially at low rates. This gives us a theoretical indication that the use of dictionaries with scales can

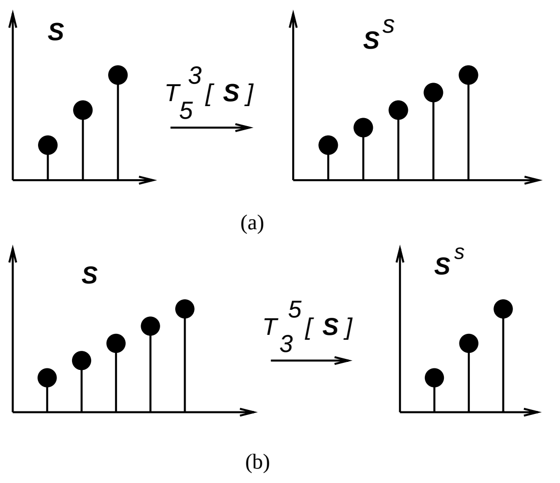


Fig. 1. Scale transformation $\mathbf{S}^s = T_N^M[\mathbf{S}]$: (a) $N > M$; (b) $N < M$.

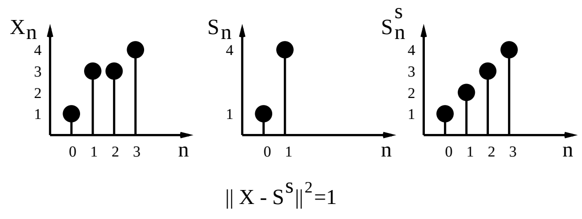


Fig. 2. Approximate pattern matching with scales.

improve performance. In what follows, we propose the MMP compression algorithm, that is based on *approximate matching with scales of recurrent patterns.*

In MMP, an input vector $\mathbf{X}$ is parsed in L non-overlapping segments and each segment is represented by a *dilated/contracted* version of a vector in a dictionary $\mathscr{D}$. The dictionary is adaptively built while the data is encoded and it is regularly updated with concatenations of previously occurred patterns. This updating rule is inspired on the workings of the Lempel–Ziv [25] lossless compression algorithm. Previous works on the extension of the Lempel–Ziv algorithm to lossy compression were made where the lossless string matching was replaced by approximate pattern matching. We call these extensions Lossy–Lempel–Ziv (LLZ) algorithms. In [3,6,10], some variations of LLZ are explored. The theoretical performance of these algorithms is analyzed in [16] and [24] where it is shown that they are not asymptotically optimal in a rate–distortion sense. However, it should be pointed out that asymptotic optimality is desirable but not essential for an algorithm to be of practical importance. In [15] an asymptotically optimal LLZ is proposed, but it is unclear if it can be implemented in practice and perform well for sources other than the binary source. Also, the performance of an asymptotically optimal algorithm can be poor for input sequences of finite blocklength. When applied to lossy image compression, these methods proved to be competitive in the range above one bit per pixel, when compared to the transform coder JPEG [19]. In [2], a two-dimensional LLZ version is developed and applied to image compression. Although achieving better performance than its one-dimensional counterparts, its performance is inferior to that of JPEG at rates below one bit per pixel. There are also related works on adaptive vector quantization [11,12]. Our algorithm proposes a new approach, differing from these previous works in both the segmentation method and the dictionary-updating procedure, not to mention the introduction of the multiscale approach. In fact, we believe that the good performance at rates below one bit per pixel, in contrast with other LLZ and adaptive VQ schemes, was in a great extent due to the use of approximate pattern matching with scales.

The simplest version of MMP is controlled by a target distortion parameter d^*. In this version of MMP, we attempt to approximate the input vector $\mathbf{X}$ using the best dictionary element as $\hat{\mathbf{X}} = T_N[\mathbf{S}_{i_0}]$. If the approximation fails, that is, the distortion $d(\mathbf{X}, \hat{\mathbf{X}})$ is above Nd^*, we split $\mathbf{X}^0 = \mathbf{X}$ in two segments $\mathbf{X}^1$ and $\mathbf{X}^2$, each of length $N/2$, and then we try to approximate each one using scaled versions of the vectors in the dictionary. If the attempt to represent any of the segments using the current dictionary fails, that segment will be split in two and the procedure repeated. This process is illustrated in Fig. 3.

Now that we have described how, given a dictionary, a segment is encoded in MMP, we need to describe how the dictionary is updated. In order to do this, we should note that there is a segmentation tree $\mathscr{S}$ associated to a given input vector $\mathbf{X}$ and a given target distortion d^*. Fig. 4 shows the segmentation three $\mathscr{S}$ associated to the segmentation in Fig. 3. Each node n_j of $\mathscr{S}$ is associated to a segment $\mathbf{X}^j$ of the input vector. The length of the vectors corresponding to nodes at depth p is $2^{-p}N$. A node n_j of the segmentation tree has either two children, nodes n_{2j+1} and n_{2j+2}, or no

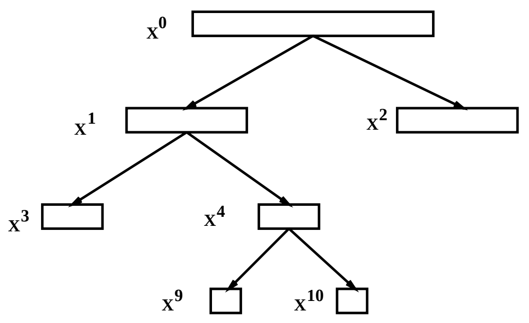


Fig. 3. Segmentation on MMP.

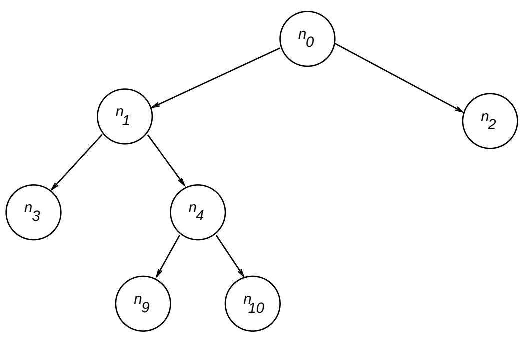


Fig. 4. Segmentation tree.

child at all. A node that has no child is a leaf node. In MMP only the leaf nodes of the segmentation tree are associated to vectors in the dictionary and are used to approximate the input vector.

For example, the segmentation tree in Fig. 4 corresponds to the segmentation of the input vector $\mathbf{X}$ in four segments, $(\mathbf{X}^3\mathbf{X}^9\mathbf{X}^{10}\mathbf{X}^2)$. In this example, each segment will be approximated by scaled versions of vectors in the dictionary, leading to the representation $\hat{\mathbf{X}} = (\hat{\mathbf{X}}^3\hat{\mathbf{X}}^9\hat{\mathbf{X}}^{10}\hat{\mathbf{X}}^2)$.

The dictionary in MMP is updated as follows: whenever the approximations $\hat{\mathbf{X}}^{2j+1}$ and $\hat{\mathbf{X}}^{2j+2}$ associated to the children nodes n_{2j+1} and n_{2j+2} are available, MMP forms an estimate of segment $\hat{\mathbf{X}}^j$, associated to the parent node n_j, as the concatenation of the approximations to the segments associated to the two children nodes. In the example of Fig. 3, when $\hat{\mathbf{X}}^9$ and $\hat{\mathbf{X}}^{10}$ are available we can concatenate them to get a new approximation $\hat{\mathbf{X}}^4$. This new approximation can then be included in the dictionary.

Therefore, in MMP, we just need to output to the decoder the information regarding the segmentation tree and the dictionary indexes corresponding to the approximations at the leaf nodes. That is, the algorithm outputs an integer sequence i_m consisting of the dictionary indexes and a sequence of binary flags b_n that specify the segmentation tree $\mathscr{S}$. The sequence of flags represent $\mathscr{S}$ as a series of binary decisions, in a top-down fashion. We use the binary flag 0 to indicate splitting and the flag 1 to indicate a leaf node. For example, the tree in Fig. 4 is represented by the sequence of flags 0,0,1,0,1,1,1.

At this point, two important aspects should be pointed out:

(1) Although the MMP uses multiscale dictionaries, the scale of a dictionary element used to encode a segment does not have to be transmitted, since it can be inferred from the segmentation tree.
(2) The dictionary is updated by the encoder using information that can be derived from just the encoded segmentation flags and dictionary indexes. Therefore, the decoder needs no side information to update the dictionary.

3. R–D Optimization of the segmentation tree

The segmentation tree generated by the distortion-controlled version of MMP is created using local decisions based on distortions calculations, and is thus not globally optimum in a rate–distortion sense. We can improve the R–D performance of MMP by optimizing its segmentation tree [9].

Referring to Figs. 3 and 4, each node n_j is associated to a segment of the input vector $\mathbf{X}^j$ that is best represented by a scaled version of a dictionary element $\mathbf{S}_{i_j}$, referred to as $\mathbf{S}^s_{i_j}$. Therefore, we can associate to each node the distortion

$$D(n_j) = d(\mathbf{X}^j, \mathbf{S}^s_{i_j}). \tag{1}$$

We call $R(n_j)$ the rate needed to specify the index i_j, that is

$$R(n_j) = -\log_2(\Pr(i_j)), \tag{2}$$

where $\Pr(i_j)$ is the probability of occurrence of index i_j in the dictionary.

The overall distortion is then

$$D(\mathscr{S}) = \sum_{n_j \in \mathscr{S}_{\mathscr{L}}} D(n_j), \tag{3}$$

where $\mathscr{S}_{\mathscr{L}}$ is the set of leaf nodes of $\mathscr{S}$. The amount of bits needed to encode this approximation is the rate $R(\mathscr{S})$, which is given by

$$R(\mathscr{S}) = R_t(\mathscr{S}) + \sum_{n_j \in \mathscr{S}_{\mathscr{L}}} R(n_j), \tag{4}$$

where $R_t(\mathscr{S})$ is the rate required to specify the segmentation tree.

The best segmentation $\mathscr{S}^*$, in an R–D sense, leads to the minimum rate $R(\mathscr{S})$ given that the distortion $D(\mathscr{S})$ is no greater than a target distortion D^* or, alternatively, the minimum distortion $D(\mathscr{S})$ at rate at most R^*. This is a constrained minimization problem stated as

$$\begin{aligned} \mathscr{S}^* &= \arg \min_{\mathscr{S} \in \mathscr{S}_{R^*}} D(\mathscr{S}), \\ \mathscr{S}_{R^*} &= \{\mathscr{S} : R(\mathscr{S}) \leqslant R^*\}. \end{aligned} \tag{5}$$

To find $\mathscr{S}^*$ we can find the solution to a related unconstrained problem introducing a Lagrange multiplier λ. It is well known that if we find the minimum of the Lagrangian cost $J(\mathscr{S}) = D(\mathscr{S}) + \lambda R(\mathscr{S})$ [4], we also find the solution to the constrained problem when we choose $R(\lambda) = R^*$. Hence, we have that

$$\begin{aligned} \mathscr{S}^* &= \arg\min_{\mathscr{S}} J(\mathscr{S}) \\ &= \arg\min_{\mathscr{S}} \left(\sum_{n_j \in \mathscr{S}_{\mathscr{L}}} D(n_j) \right) \\ &\quad + \lambda \left(R_t(\mathscr{S}) + \sum_{n_j \in \mathscr{S}_{\mathscr{L}}} R(n_j) \right) \\ &= \arg\min_{\mathscr{S}} \lambda R_t(\mathscr{S}) + \sum_{n_j \in \mathscr{S}_{\mathscr{L}}} (D(n_j) + \lambda R(n_j)) \\ &= \arg\min_{\mathscr{S}} \lambda R_t(\mathscr{S}) + \sum_{n_j \in \mathscr{S}_{\mathscr{L}}} J(n_j), \end{aligned} \tag{6}$$

where $J(n_j) = D(n_j) + \lambda R(n_j)$.

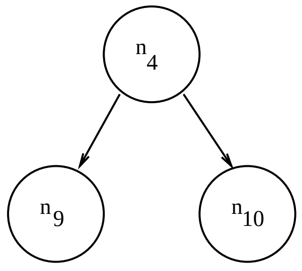


Fig. 5. The sub-tree $\mathscr{S}(n_4)$ of the binary tree in Fig. 4.

A sub-tree $\mathscr{S}(n_j)$ of $\mathscr{S}$ at node n_j is the binary tree composed by all the nodes of $\mathscr{S}$ that have n_j as the root node. Fig. 5 illustrates the sub-tree $\mathscr{S}(n_4)$ of the binary tree in Fig. 4. We denote $\mathscr{S} - \mathscr{S}(n_j)$ the tree obtained from $\mathscr{S}$ by pruning the sub-tree $\mathscr{S}(n_j)$.

If the Lagrangian costs $J(n_l)$, associated with the approximation of each segment $\mathbf{X}'$, are independent, then the Lagrangian cost of two sub-trees $J(\mathscr{S}(n_l))$ and $J(\mathscr{S}(n_m))$ are also independent, as long as all nodes of both sub-trees are different. Then a fast search algorithm, similar to [21], can be implemented considering that if $J(n_l) \leqslant J(\mathscr{S}(n_{2l+1})) + J(\mathscr{S}(n_{2l+2}))$ then the sub-trees $\mathscr{S}(n_{2l+1})$ and $\mathscr{S}(n_{2l+2})$ must be pruned from $\mathscr{S}$ in order to decrease the cost. Unfortunately, this is not the case with MMP, since the costs $J(n_l)$ are coupled by the dictionary updating procedure. That is, the computation of the overall variation of the cost obtained by pruning subtree $\mathscr{S}(n_l)$ has to take into account the variation in the cost of the other nodes $J(n_k)$ due to the variation of the dictionary (that was caused by the pruning of sub-tree $S(n_l)$). However, if the initial dictionaries are large enough, one can argue that the effect of the dictionary update on the minimization of $J(n_l)$ can be negligible. In practical MMP implementations, we usually impose an upper limit to the size M of the input vector $\mathbf{X}$ due to the finite amount of memory available. This forces the input data to be broken in blocks of size M that are sequentially processed by MMP algorithm. (Note that the initial dictionary of one block is the final dictionary of the previous block.) Although the assumption of a large initial dictionary is not true for the first blocks, the dictionary eventually grows large enough so that the Lagrangian costs $J(n_l)$ are almost decoupled. In this sense, one could use the algorithm in [21] to find an approximate R–D-optimal solution.

However, if we want to use relatively large block-lengths or if the dictionary is too small (as happens at very low rates), we should modify the algorithm to take into account the impact of the dictionary-updating procedure. We know that the dictionary is updated by the inclusion of the concatenation of previously encoded segments. Therefore, if we choose to prune a sub-tree, the impact in the cost is not restricted to that sub-tree, but can affect all nodes that are to the right of the sub-tree. That is, if we prune a sub-tree, we might remove from the dictionary an element that would otherwise be used later to approximate an input segment, therefore increasing the cost. The idea is to prune a sub-tree only if the potential increase in the cost of subsequent nodes, due to the removal of some vectors from the dictionary, is not greater than the reduction in the cost provided by the pruning. The algorithm for encoding a length-M segment is described below. Note that the children of node n_j are nodes n_{2j+1} and n_{2j+2}.

Step 1: Initialize $\mathcal{S}$ as the full tree of depth $\log_2(M)+1$.

Step 2: Make $J_i=\infty$ for the M leaf nodes, that is, for $i=M-1,M,\ldots,2M-2$.

Step 3: Make $p=\log_2(M)$ and $\mathcal{S}_0=\mathcal{S}$.

Step 4: For each node $n_l\in\mathcal{S}$ at depth p, that is, for $l\in\{2^{p-1}-1,2^{p-1},\ldots,2^p-2\}$, evaluate:

(i) $J_l=J(n_l)+\lambda R_{1_l}$, where $J(n_l)$ is the cost to represent the input segment associated to the node n_l and R_{1_l} is the rate needed to indicate that the node n_l is a leaf.

(ii) $\Delta J_l=\sum_{n_r\in\mathcal{S}-\mathcal{S}(n_l)}J(n_r)-\sum_{n_r\in\mathcal{S}-\mathcal{S}(n_l)}J'(n_r)$, where $J'(n_l)$ is computed using the dictionary without $\hat{\mathbf{X}}^{l}=(\hat{\mathbf{X}}^{2l+1}\hat{\mathbf{X}}^{2l+2})$, that is, the dictionary that would be obtained without the sub-tree $\mathcal{S}(n_l)$.

Step 5: If $J_l-J_{2l+1}-J_{2l+2}-\lambda R_{0_l}\leqslant\Delta J_l$ then prune nodes n_{2l+1} and n_{2l+2} from $\mathcal{S}$. (R_{0_l} is the rate needed to indicate splitting, and J_{2l+1},J_{2l+2} were evaluated in the previous iteration with $p+1$). Otherwise, the cost of node n_l is updated with $J_{2l+1}+J_{2l+2}+\lambda R_{0_l}$.

Step 6: Make $p=p-1$.

Step 7: Repeat Steps 4–6 until $p=0$.

Step 8: If $\mathcal{S}=\mathcal{S}_0$ then the optimization is done. Otherwise, go to Step 3.

It is interesting to consider why ΔJ_l is evaluated using all the nodes not belonging to $\mathcal{S}(n_l)$, since only the leaf nodes contribute to the total cost. The idea of the extended algorithm is to prune a subtree only when we are sure that the cost will not increase. The computation of ΔJ_l must be conservative because we do not know, at the time we are making a decision at node n_l, which nodes will be the leaves (the current leaves may be pruned later). That is, when we evaluate $\Delta J_l<J_l-J_{2l+1}-J_{2l+2}-\lambda R_{0_l}$, we only prune the sub-tree $\mathcal{S}(n_l)$ when we are sure, irrespective of which will be the final tree $\mathcal{S}-\mathcal{S}(n_l)$, that the cost will decrease. If the local cost increases with the pruning, and we decide not to prune, this decision might either be the correct one or not, depending on the future tree $\mathcal{S}-\mathcal{S}(n_l)$. This is the reason why we iterate the algorithm until convergence. Note that convergence is guaranteed because the algorithm ensures that the cost does not increase in any iteration.

4. MMP with multidimensional sources

One can view the segmentation process of MMP as splitting the input vector $\mathbf{X}^j$ in two segments around the segmentation point $p_s=\ell(\mathbf{X}^j)/2$, where $\ell(\mathbf{X}^j)$ is the length of $\mathbf{X}^j$. In this case p_s is a point in a one-dimensional space. The segmentation procedure in MMP can be easily extended to operate with multidimensional sources, if we consider a point $\mathbf{p}_s$ defined in a multidimensional space. For example, in the two-dimensional case, the input vector is replaced by an input *matrix* $\mathbf{X}$ of size $N\times N$. In this case, there are many possibilities to choose the segmentation point $\mathbf{p}_s=(p_l\ p_c)$. If we choose $\mathbf{p}_s=(\ell_{\text{row}}(\mathbf{X}^j)/2\ \ \ell_{\text{col}}(\mathbf{X}^j)/2)$, where $\ell_{\text{row}}(\mathbf{X}^j)$ is the number of rows of the matrix $\mathbf{X}^j$ and $\ell_{\text{col}}(\mathbf{X}^j)$ is its number of columns, then the matrix $\mathbf{X}^j$ is sub-divided in four sub-matrices as $\mathbf{X}^j=\left(\begin{smallmatrix}\mathbf{X}^{4j+1} & \mathbf{X}^{4j+2}\\ \mathbf{X}^{4j+3} & \mathbf{X}^{4j+4}\end{smallmatrix}\right)$. This choice of $\mathbf{p}_s$ leads to a *quad-tree*-segmentation. This segmentation can be represented by a quaternary tree where a node n_j can have four children, $n_{4j+1},n_{4j+2},n_{4j+3},n_{4j+4}$, or no child at all. A node n_j at depth p is associated to a matrix $\mathbf{X}^j$ of size $2^{-p}N\times 2^{-p}N$.

If we want to keep the binary tree representation, we can use different segmentation points. For example, one possible rule is: if $\ell_{\text{col}}(\mathbf{X}^j)>\ell_{\text{row}}(\mathbf{X}^j)$ then $\mathbf{p}_s=(0\ \ \ell_{\text{col}}(\mathbf{X}^j)/2)$, otherwise $\mathbf{p}_s=(\ell_{\text{row}}(\mathbf{X}^j)/2\ \ 0)$. This choice of segmentation points leads to the segmentation illustrated in Fig. 6.

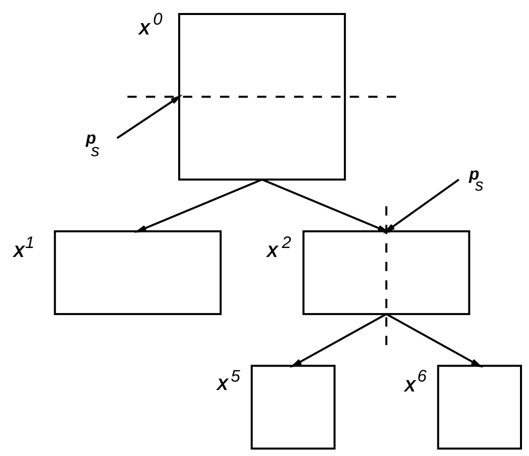


Fig. 6. A two-dimensional segmentation example.

In other words, if $\ell_{\text{col}}(\mathbf{X}^j) > \ell_{\text{row}}(\mathbf{X}^j)$ then the matrix $\mathbf{X}^j$ is split as $\mathbf{X}^j = (\mathbf{X}^{2j+1}\mathbf{X}^{2j+2})$, otherwise as $\mathbf{X}^j = \binom{\mathbf{X}^{2j+1}}{\mathbf{X}^{2j+2}}$. This segmentation can be represented by a binary tree, and a node n_j at depth p is associated to a matrix $\mathbf{X}^j$ of size $(2^{-\lfloor (p+1)/2 \rfloor}N \times 2^{-\lfloor p/2 \rfloor}N)$, where $\lfloor x \rfloor$ is the largest integer less than or equal to x. In our experiments, this binary segmentation leads to better performance than the quadtree one.

The two-dimensional MMP must use a two-dimensional scale transformation $T_{N,M}^{N',M'}: \mathbb{R}^{N'M'} \mapsto \mathbb{R}^{NM}$ in order to make two-dimensional approximate pattern matching with scales. This scale transformation maps a matrix of dimensions $N' \times M'$ to a matrix of dimensions $N \times M$. This whole procedure can be trivially extended to more than two dimensions. Similarly to the one-dimensional case, we can use the simplified notation $T_{N,M}\lfloor \mathbf{X}^j \rfloor$ implying that $\mathbf{X}^j$ is of size $N' \times M'$.

5. Multiple-codebook implementation of MMP

When MMP tries to represent an input segment $\mathbf{X}^j$ by one of the vectors in its dictionary $\mathscr{D}$, it first has to apply scale transformations to adjust the length of each vector in $\mathscr{D}$ to the same as the length of $\mathbf{X}^j$. If the length of the input vector is N, the segmentation procedure can create vectors of lengths $N/2, N/4, \ldots, 1$. This implies that there are at most $1 + \log_2(N)$ different lengths or scales. Therefore, to save time, we could keep $1 + \log_2(N)$ copies of the dictionary, one at each scale, to avoid the computation of the scale transformation each time we want to do a match. This way, we only need to use the scale transformation when we are including a new vector in the dictionary. We denote a copy of the dictionary at scale $2^{-2p}N$ as $\mathscr{D}^p$. If we are using this multiple-codebook scheme, to include a new vector $\hat{\mathbf{X}}^j$ in the dictionary we must actually include $T_{2^{-2p}N}[\hat{\mathbf{X}}^j]$ in $\mathscr{D}^p$ for $p = 0, 1, \ldots, \log_2(N)$.

6. Experimental results

We have implemented the distortion-controlled MMP and the rate-distortion optimized MMP, both with the two-dimensional segmentation (binary tree) described in Section 4. We call these implementations 2D-MMP and 2D-MMP-RD, respectively. We used the programs to lossy compress gray-scale still image data. The input images were initially divided in $N \times N$ blocks that were sequentially processed by the algorithms. We used the multiple-codebook approach with $1 + 2\log_2(N)$ codebooks. The initial dictionary at scale 1×1 was $\mathscr{D}_0^0 = \{-128, -124, \ldots, -4, 0, +4, \ldots, +124\}$. The initial dictionaries $\mathscr{D}_0^p$ at all other scales $(2^{-\lfloor (p+1)/2 \rfloor}N \times 2^{-\lfloor p/2 \rfloor}N)$, $p = 0, 1, \ldots, 2\log_2(N)$ were obtained from $\mathscr{D}_0^0$ by the use of a two-dimensional scale transformation. We have chosen to use a *separable* scale transformation based on a one-dimensional transformation that was applied independently to the rows and to the columns of $\mathbf{X}^j$. The one-dimensional scale transformation was implemented using classical sampling-rate change operations. When we want to change from length N_0 to length N, with $N > N_0$, we first change to length N_0N using a linear interpolator as the filter. Then we make a downsampling by N_0 operation. The whole procedure is defined in Eq. (7).

$$m_n^0 = \left\lfloor \frac{n(N_0 - 1)}{N} \right\rfloor,$$

$$m_n^1 = \begin{cases} m_n^0 + 1, & m_n^0 < N_0 - 1, \\ m_n^0, & m_n^0 = N_0 - 1, \end{cases}$$

$$\alpha_n = n(N_0 - 1) - Nm_n^0,$$

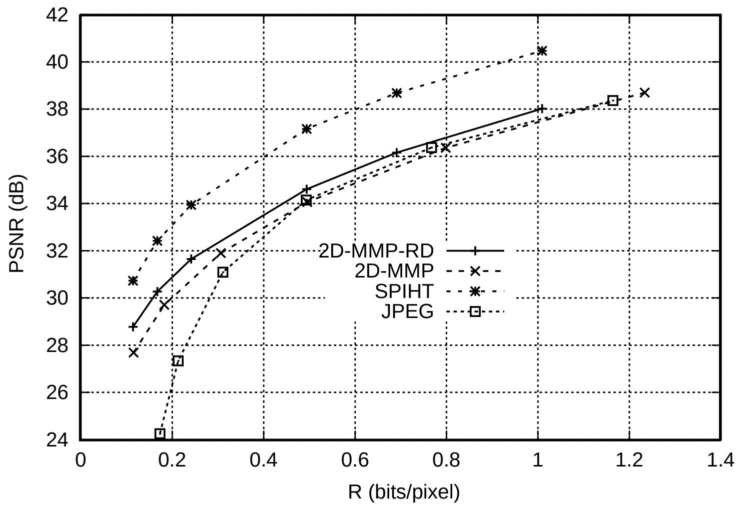


Fig. 7. R–D performance with LENA 512 × 512.

$$S^s_n = \left\lfloor \frac{\alpha_n(S_{m^1_n} - S_{m^0_n})}{N} \right\rfloor + S_{m^0_n},$$

$$n = 0, 1, \ldots, N-1. \tag{7}$$

When $N_0 > N$, we first change to length N_0N using a linear interpolator as the filter and then we apply a mean filter to length N_0+1, followed by a downsampling by N_0 operation. The procedure is defined in Eq. (8).

$$m^0_{n,k} = \left\lfloor \frac{n(N_0-1)+k}{N} \right\rfloor,$$

$$m^1_{n,k} = \begin{cases} m^0_{n,k}+1, \; m^0_{n,k} < N_0 - 1, \\ m^0_{n,k}, \qquad m^0_{n,k} = N_0 - 1, \end{cases}$$

$$\alpha_n = n(N_0-1) + k - Nm^0_n,$$

$$S^s_n = S_{m^0_{n,k}} + \frac{1}{N_0+1}\sum_{k=0}^{N_0} \left\lfloor \frac{\alpha_{n,k}(S_{m^1_{n,k}} - S_{m^0_{n,k}})}{N} \right\rfloor,$$

$$n = 0, \ldots, N-1. \tag{8}$$

The two-dimensional separable scale transformation was implemented as in Eq. (9)

$$\mathbf{Y}_i = T_M[\mathbf{S}_i], \quad i = 0, 1, \ldots, \ell(\mathbf{S}_i) - 1,$$
$$\mathbf{S}^s_j = \left(T_N\left[(\mathbf{Y}^{\mathrm{T}})_j\right]\right)^{\mathrm{T}}, \quad j = 0, 1, \ldots, M-1, \tag{9}$$

where we use $\mathbf{X}_i$ to denote the rows of $\mathbf{X}$. The sequence of dictionary indexes i_m, as well as the sequence of binary flags b_n were encoded using an adaptive arithmetic encoder with different contexts for each scale.

Figs. 7–10 show the R–D performance of the algorithms with the images Lena, F-16, PP1205 and PP1209, all of dimensions 512 × 512. In all cases, the images were divided into 8 × 8 blocks. The results for the algorithm SPIHT [20] and JPEG [19] are also shown for comparison. The popular images Lena and F-16 were downloaded from the location http://www.sipi.usc.edu. The image PP1205 was scanned from page 1205 of the *IEEE Transactions on Image processing*, volume 9, number 7, July 2000. It contains text and mathematical formulas. The image PP1209 was scanned from page 1209 of the same magazine and is a compound of gray-scale (two compressed versions of Lena), text, formulas and graphics.

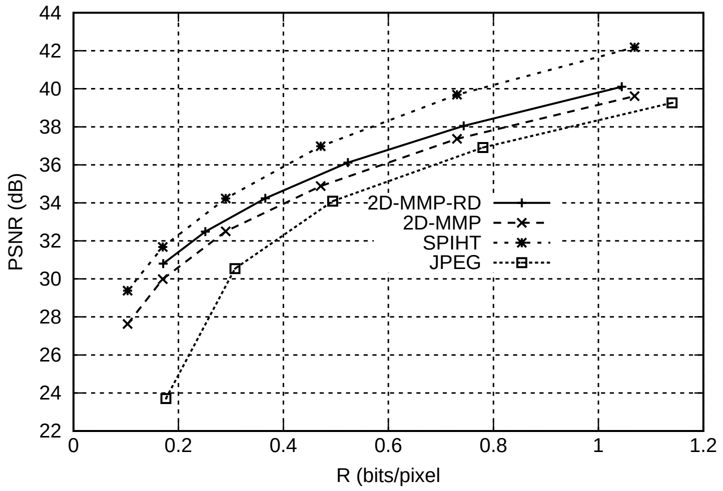


Fig. 8. R–D performance with F-16 512 × 512.

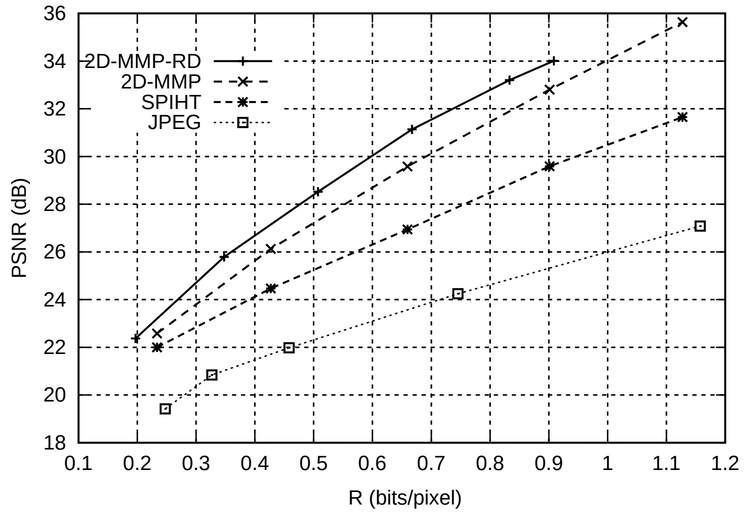


Fig. 9. R–D performance with PP1205 512 × 512.

The figures show us that:

(i) The 2D-MMP-RD algorithm outperformed 2D-MMP for all test images.

(ii) The 2D-MMP-RD algorithm outperformed SPIHT for the pure text and the compound (by ≈ 4.4 dB for PP1205 and ≈ 1.0 dB for PP1209).

(iii) The SPIHT algorithm outperformed 2D-MMP-RD for the pure grayscale images (by ≈ 2.5 dB for Lena and ≈ 1.9 dB for F-16).

(iv) The 2D-MMP-RD algorithm outperformed JPEG for all test images.

Figs. 11–13 show some test images compressed by 2D-MMP-RD and SPIHT at 0.50 bits/pixel. We can see some blocking effects in the image Lena that are absent in the wavelet-based SPIHT. In fact, the specific DWT used, based on the 9–7 biorthogonal set of linear phase FIR filters of [1], has long basis vectors which help to eliminate the blockiness. However, this

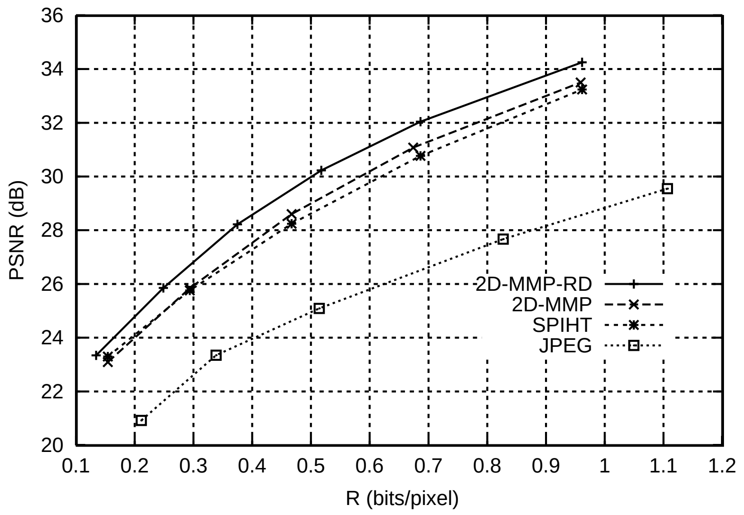


Fig. 10. R–D performance with PP1209 512 × 512.

same property becomes a drawback when we try to approximate signals with many edges, like in PP1205 and PP1209. Note that the performance of MMP in these two cases is very good. It is also important to point out that, although the MMP performance was below the one of SPIHT for typical gray-scale images, it outperformed JPEG. In addition, the fact that it performed very well for text, graphics and compound images, indicates the MMP's dictionary updating procedure lends to MMP a universal flavor. This gains relevance specially if one considers that the initial dictionaries were the same for all images, with $\mathscr{D}_0^0 = \{-128, -124, \ldots, -4, 0, +4, \ldots, +124\}$. One drawback of the presented method, however, is the blockiness of the reconstructed images. In the next section we address the blocking effect control in MMP.

7. Blocking effect control

MMP is based on a segmentation procedure that parses the input vector $\mathbf{X}$ in L segments $\mathbf{X} = (\mathbf{X}^{l_0} \cdots \mathbf{X}^{l_{L-1}})$. The procedure can guarantee that the mean distortion in the approximation of each segment $\hat{\mathbf{X}}^{l_m}$, and therefore in the concatenation $\hat{\mathbf{X}}$, is less than or equal to a target distortion d^*. Alternatively, the segmentation tree can be optimized to minimize $D + \lambda R$. However, in either case the algorithms have no control regarding the smoothness of the approximation at the boundaries of the segments, unless the target distortion d^* or the value of the λ parameters is zero. This gives rise to blocking effects. The concatenation of two segments can alternatively be viewed as the sum of two non-overlapping vectors $\hat{\mathbf{X}}_s^{l_m} = (\mathbf{X}^{l_m}\ \mathbf{0}_{\ell(\mathbf{X}^{l_{m+1}})})$ and $\hat{\mathbf{X}}_s^{l_{m+1}} = (\mathbf{0}_{\ell(\mathbf{X}^{l_m})}\ \mathbf{X}^{l_{m+1}})$. One way to improve the smoothness at the boundary, and in consequence reduce the blocking effects, is to allow overlapping of the segments. However, if we use overlapping vectors at the encoder, we can no longer guarantee that the sum of two segments, each one with mean distortion below the target d^* will have mean distortion below d^*. We have made experiments using overlapping vectors in the dictionaries of both the encoder and the decoder, and found it difficult to optimize the choice of bases at the encoder side.

To circumvent the difficult optimization problem posed by the use of overlapped vectors at the encoder side, we can use non-overlapped vectors to encode the input data, and a different set of overlapping vectors to reconstruct the data at the decoder side [8]. One could make an analogy with decomposition/reconstruction of vectors using biorthogonal bases, where the coefficients of the decomposition are computed using a set of non-overlapping basis vectors at the encoder and the reconstruction is carried out using another

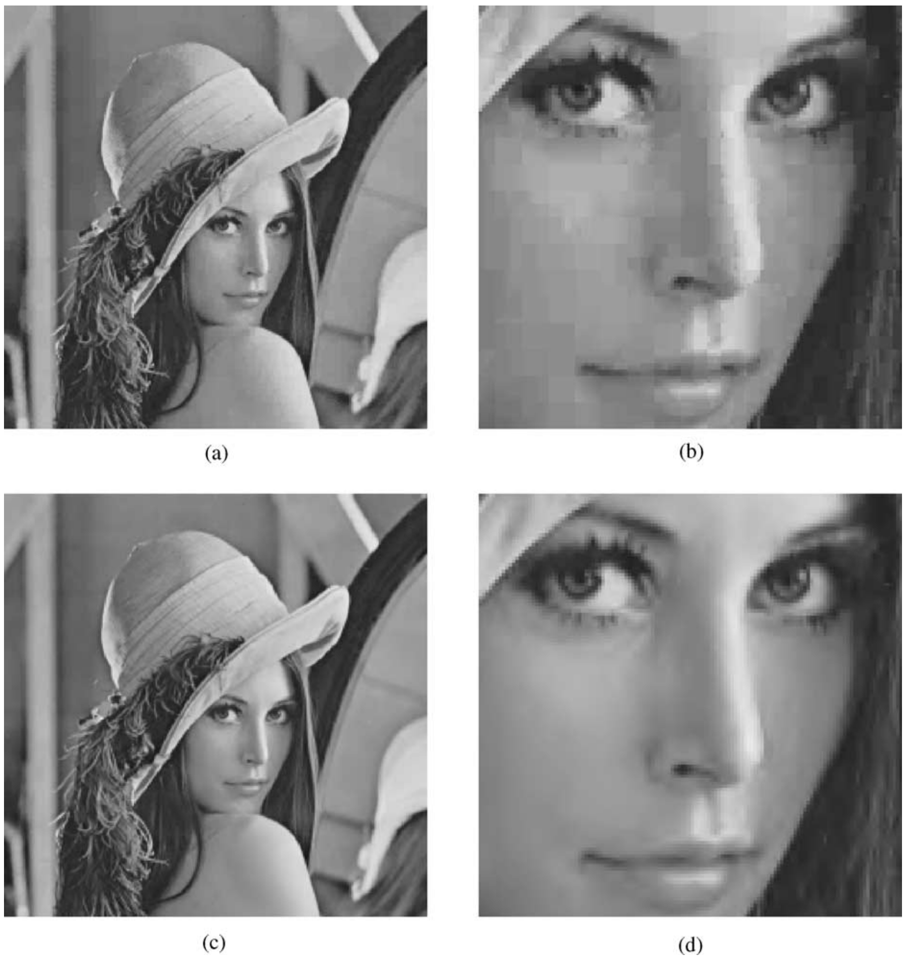


Fig. 11. Image Lena: (a) 2D-MMP-RD at 0.5 bpp. PSNR = 34.7 dB; (b) detail; (c) SPIHT at 0.50 bpp. PSNR = 37.2 dB; (d) detail.

set of overlapping basis vectors at the decoder. For example, let us consider that MMP parses the input vector $\mathbf{X}$ in four segments of same length and approximates each one by an equal-components vector. That is, the segmentation tree is a binary tree of depth 2 and $\mathbf{X} = (\mathbf{X}^3\mathbf{X}^4\mathbf{X}^5\mathbf{X}^6)$. The approximation of each segment by an equal-components vector yields $\hat{\mathbf{X}} = (a_0\mathbf{1}_{N/4}\ a_1\mathbf{1}_{N/4}\ a_2\mathbf{1}_{N/4}\ a_3\mathbf{1}_{N/4})$. The encoder has then the task of reconstructing the input vector from the four dictionary indexes representing these equal-components vectors. This is analog to recovering an approximation to $\mathbf{X}$ from a down-sampled by $N/4$ version of it, $\hat{\mathbf{X}}_d = (a_0a_1a_2a_3)$ by filtering its interpolated by $N/4$ version using a filter with constant impulse response, $\mathbf{1}_{N/4}$. It is well known from multirate filter banks theory [23] that we can recover a lowpass version of $\mathbf{X}$ from its down-sampled version using a synthesis filter that can be different from the analysis filter (in this case, $\mathbf{1}_{N/4}$). So we can tailor the impulse response of the synthesis filter, or, on the MMP decoder, the corresponding basis vector, to match a smoothness requirement.

We used an adaptive FIR post-filtering technique to modify the shape of the reconstruction vectors at the decoder, as follows:

- First, we find an approximation for $\mathbf{X}$ using the standard decoder, $\hat{\mathbf{X}} = (\mathbf{s}_{i_0}^{(k_0)}\mathbf{s}_{i_1}^{(k_1)}\cdots\mathbf{s}_{i_{L-1}}^{(k_{L-1})})$. That is, $\hat{\mathbf{X}}$ is

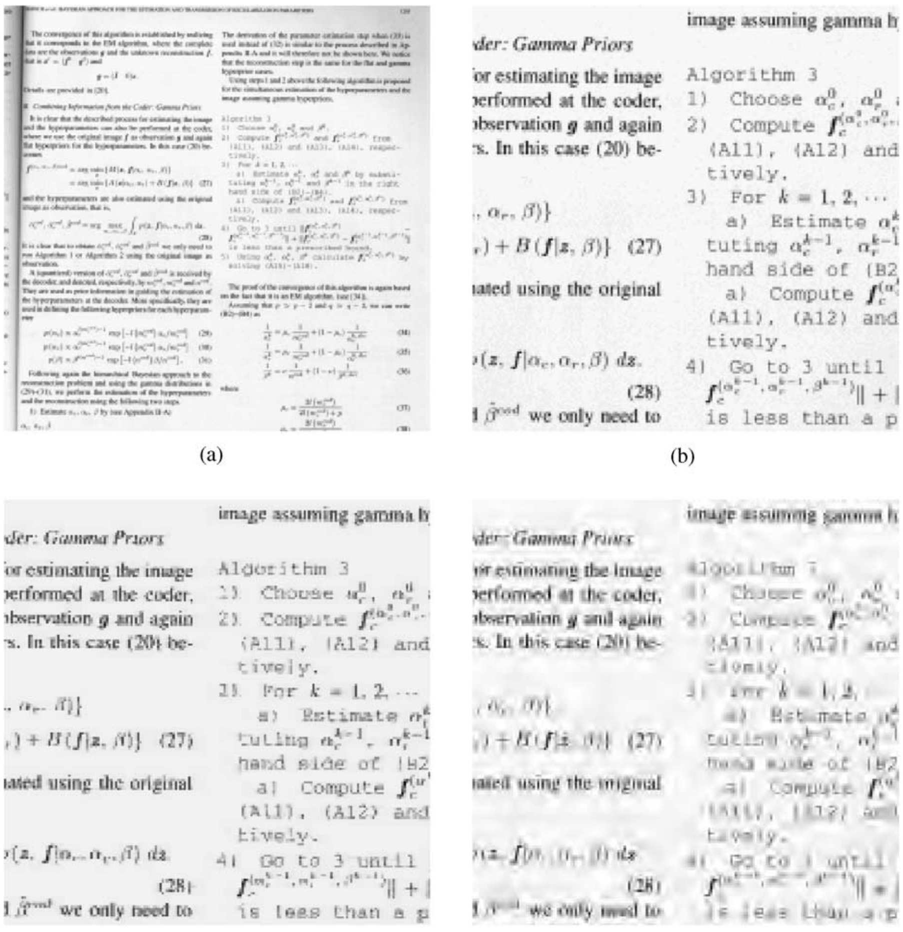
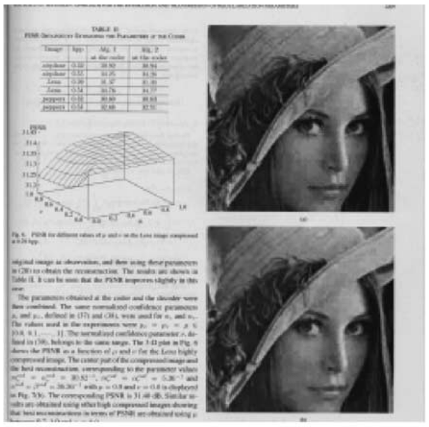


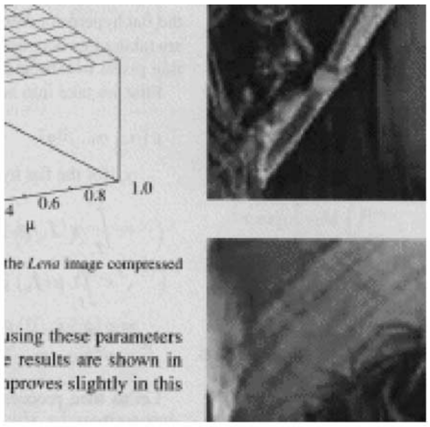
Fig. 12. Image PP1205: (a) original; (b) detail; (c) 2D-MMP-RD at 0.5 bpp. PSNR = 28.5; (d) SPIHT at 0.50 bpp. PSNR = 25.2 dB.

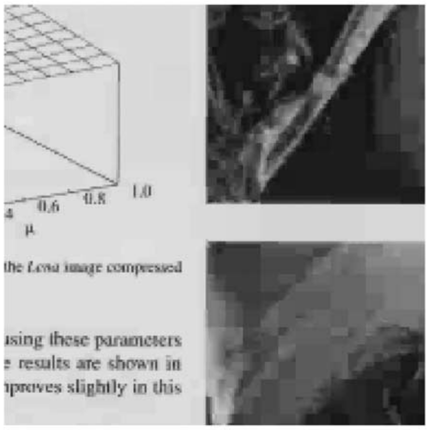
composed by the concatenation of L segments, each one a vector $\mathbf{s}_{i_m}^{(k_m)}$ in the dictionary of scale k_m.

- Next, we replace each $\mathbf{s}_{i_m}^{(k_m)}$ by a concatenation of vectors in the initial dictionaries. That is, $\hat{\mathbf{X}} = (\mathbf{s}_{i'_0}^{(k'_0)}\mathbf{s}_{i'_1}^{(k'_1)}\cdots\mathbf{s}_{i'_{q-1}}^{(k'_{q-1})})$, where the vectors $\mathbf{s}_{i'_m}^{(k'_m)}$ are in the initial dictionaries $\mathscr{D}_0^{(k)}$ (that are composed of constant functions). This creates a new segmentation of the reconstructed vector $\hat{\mathbf{X}}$, where each segment corresponds to an element of the initial dictionaries, $\hat{\mathbf{X}} = (\hat{\mathbf{X}}^{l_0}\hat{\mathbf{X}}^{l_1}\cdots\hat{\mathbf{X}}^{l_{q-1}})$.
- Finally, we apply a zero-delay FIR filter to $\hat{\mathbf{X}}$. The length of the filter at each segment $\hat{X}^{l_m}$ is proportional to the length of the segment $\ell(\hat{\mathbf{X}}^{l_m})$.

Therefore, the size of the filter is adjusted in a sample-by-sample basis. Considering again, for example, that the reconstructed $\hat{\mathbf{X}}$ is composed of four segments of equal-components vectors, and the filter is a moving average filter of length L_k+1, this procedure replaces the four non-overlapping flat vectors of size $N/4$ by four overlapping triangle-shaped vectors of size $N/2$. Fig. 14 illustrates the process. In this figure, the output vector is composed of three segments. The component being filtered is indicated by the arrow. As can be seen, the length of the FIR filter, a moving average filter in this case, is proportional to the size of the original reconstruction vector.


(a)


(b)


(c)

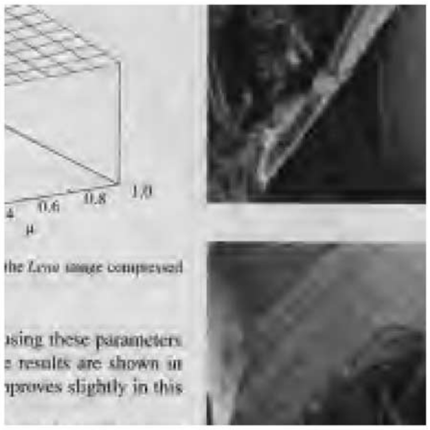

(d)

Fig. 13. Image PP1209: (a) original; (b) detail; (c) 2D-MMP-RD at 0.5 bpp.PSNR = 29.9; (d) SPIHT at 0.50 bpp. PSNR = 28.7 dB.

We have made no attempt to optimize the impulse response of the adaptive filter used at the decoder. However, we got a good compromise between the objective and the subjective performances when the impulse response of the adaptive filter is Gaussian shaped. The impulse response of this filter, for a $N \times M$ block was given by

$$h_{n,m} = \begin{cases} C_0 e^{-C_1((\frac{n}{N})^2+(\frac{m}{M})^2)}, & 1-N \leqslant n \leqslant N-1, \\ & 1-M \leqslant m \leqslant M-1, \\ 0, & \text{elsewhere,} \end{cases}$$

where

$$C_0 = \sum_{n=1-N}^{N-1} \sum_{m=1-M}^{M-1} \mathrm{e}^{-C_1((n/N)^2+(m/M)^2)}. \qquad (10)$$

Fig. 15 shows the image Lena reconstructed with this Gaussian filter and $C_0 = 6$. We can see in the detail the reduction of the blocking effect, when compared to Fig. 11. It is interesting to note that the reduction in blockiness has not been obtained at the expense of PSNR performance. In fact, the PSNR performance indeed increased by 0.2 dB. With the image F-16 there was a similar improvement but the objective

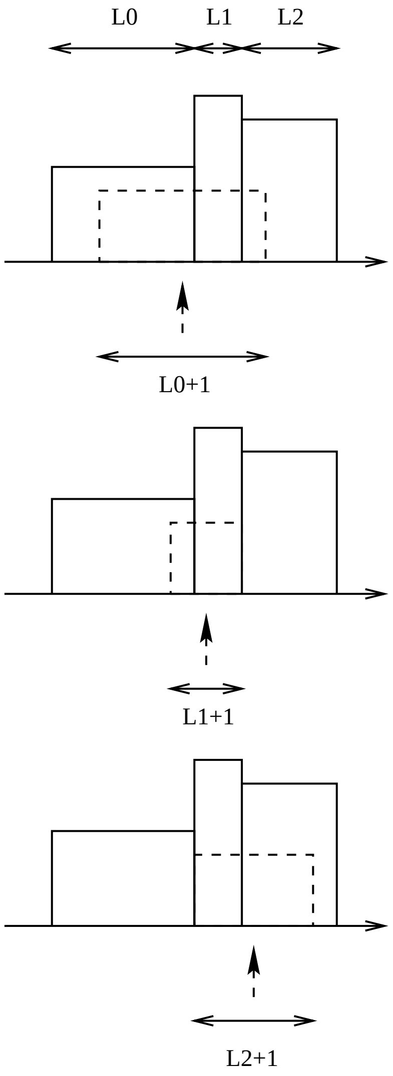


Fig. 14. FIR post-filtering process.

performance dropped 0.2 dB for the images PP1205 and PP1209. This result is not surprising, since the decoder does not perform just a blind blockiness reduction. In fact, the segmentation tree and the sequence of dictionary indexes inform the decoder the length and position of the flat-shaped vectors used to build the reconstructed signal. Therefore, they also convey to the decoder information regarding the space-frequency distribution of the energy of the input signal. The adaptive filtering procedure then adjusts the frequency response of the filter in order to keep most of the original signal energy unaffected. In brief, it uses the available information to the decoder to perform an effective adaptive filtering. This idea could be further extended in the spirit of consistent reconstructions [13].

One should note that overlapping bases tend to be good for gray-scale images, while non-overlapping bases tend to be good for text and graphics. This is confirmed by the increase in performance for Lena and F16 and for the decrease in performance for PP1205 and PP1209 when using overlapping vectors in the reconstruction.

Therefore, ideally, we would like to have a variation of the MMP algorithm that would allow the use of overlapped bases at the encoder side, and could somehow decide whether to use them or not, depending on local image content. This would certainly provide superior performance, and we consider this a very interesting topic for future research.

8. Conclusions

In this paper, we have proposed a novel method for multidimensional signal compression, based on multiscale recurrent pattern matching, the MMP algorithm.

One of the pillars of the MMP algorithm is the use of scales. That is, we can use, besides the vectors in a dictionary, all expansions and contractions of them. The input signal is then segmented into variable-size vectors and each segment is encoded using a scaled dictionary element. The segmentation can be optimized in a rate–distortion sense.

Another pillar of MMP is the dictionary updating procedure. The initial dictionary is very simple, and it is updated as the data is being encoded, using concatenations of expansions and contractions of previously encoded vectors. The dictionary updating is made in such a way that the decoder can update the dictionary without the need of any side information. Therefore, it can adapt to different kinds of sources, as gray-scale still images, motion compensated frame differences, text and graphics, and so on. In a limited sense, one can say that the MMP algorithm has a universal character.

An important characteristic of MMP is also that the signal segmentation procedure can be trivially

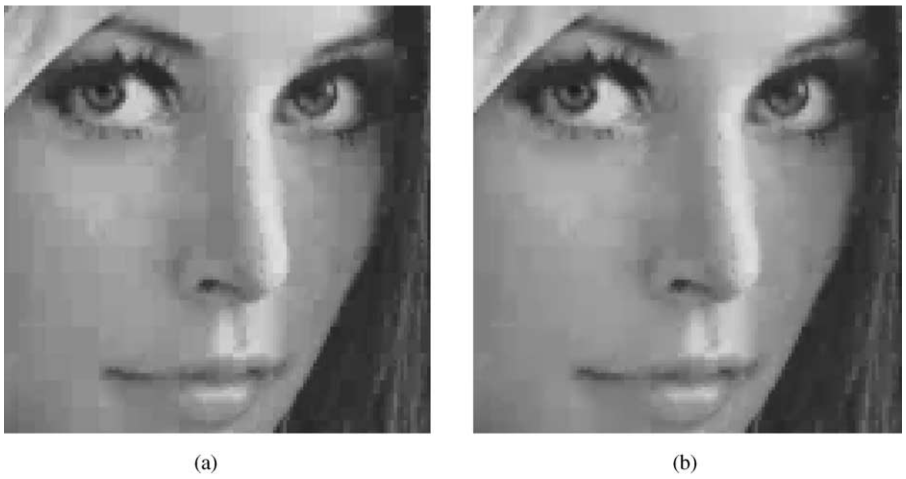


Fig. 15. Image Lena: (a) 2D-MMP-RD at 0.5 bpp.PSNR = 34.7 dB; (b) post-processed with Gaussian ($C_0 = 6$).PSNR = 34.9 dB.

carried out in multiple dimensions; therefore, MMP can be readily adapted to operate on multidimensional data.

It is also worth of notice that the MMP algorithm can be lossless, provided that the initial dictionary expands the full dynamic range of the input signal. This is so because, in the worst case, the segmentation tree will grow to full length, with its leaves corresponding to scalars. The scalars can then be lossless encoded by the words in the initial dictionary. Therefore the MMP has the potential to perform both lossy and lossless compression in a single, unified way.

We have developed a blockiness reduction procedure that uses different analysis and synthesis vectors. This approach suggests that a variation of MMP using overlapped vectors both at the encoder and decoder can lead to superior performance for a large class of image data.

It is important to mention that the MMP algorithm developed here represents just the first steps in the research of a class of signal compression algorithms based on multiscale recurrent pattern matching. The mathematical analysis made in Appendix A shows that approximate pattern matching of Gaussian vectors using different scales can be advantageous. This, in principle, gives a good indication of the potentialities of the methods developed here. This is further reinforced by the fact that its performance is already reasonably good for a large class of images. Therefore, in our opinion, research in the use of multiscale recurrent pattern matching its worth being further pursued.

Appendix A. Approximate pattern matching with memoryless Gaussian sources

In this appendix we investigate the matching properties of Gaussian vectors. Our goal is to get some insight on the workings of approximate pattern matching with scales, justifying its use in a low-rate lossy compression scheme.

A.1. Matching with a dictionary of Gaussian vectors

Let $\mathbf{x} = (x_1 x_2 \dots x_N)^{\mathrm{T}}$ be a zero mean Gaussian vector with covariance matrix $\mathbf{K}_x$ and let $\mathscr{D} = \{\mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^M\}$ be a dictionary where $\mathbf{y}^i = (y_1^i y_2^i \dots y_N^i)^{\mathrm{T}}$ are zero-mean Gaussian vectors with covariance matrix $\mathbf{K}_{y^i} = \mathbf{K}_x$. Let $\mathbf{x}$ and $\mathbf{y}^i, i \in 1, M$ be mutually independent [18].

We say a match occurs when at least one of the dictionary vectors can be used to replace $\mathbf{x}$ with distortion at most Nd^*. If the distortion measure is the mean squared error, then we have a match if:

$$\|\mathbf{x} - \mathbf{y}^i\|^2 \leqslant Nd^* \quad \text{for some } i \in [1, M]. \tag{A.1}$$

The probability of match P_m is then

$$P_m = \mathbf{1} - \Pr\{\|\mathbf{x}-\mathbf{y}^i\|^2 > Nd^* \quad \text{for all } i \in [1,M]\}. \tag{A.2}$$

Using all the M vectors of the dictionary we form a vector **u** defined by

$$\mathbf{u} = ((\mathbf{y}^1)^{\mathrm{T}}(\mathbf{y}^2)^{\mathrm{T}}\ldots(\mathbf{y}^M)^{\mathrm{T}})^{\mathrm{T}} \tag{A.3}$$

The dictionary vectors are mutually independent zero-mean Gaussian vectors with covariance matrices $\mathbf{K}_x$, so the vector **u** is zero-mean Gaussian with covariance matrix

$$\mathbf{K}_u = \mathbf{I}_M \otimes \mathbf{K}_x, \tag{A.4}$$

where $\mathbf{I}_M$ is a $M \times M$ identity matrix and $\otimes$ denotes a Kronecker product.

For a particular realization **X** of the random vector **x** we can define the difference vector r^i as.

$$\mathrm{r}^i = \mathbf{y}^i - \mathbf{X}, \quad i \in [1,M] \tag{A.5}$$

and the difference vector vector **v** as

$$\mathbf{v} = ((\mathbf{r}^1)^{\mathrm{T}}(\mathbf{r}^2)^{\mathrm{T}}\ldots(\mathbf{r}^M)^{\mathrm{T}})^{\mathrm{T}} \tag{A.6}$$

or using Eqs. (A.3) and (A.5).

$$\mathbf{v} = \mathbf{u} - \mathbf{1}_M \otimes \mathbf{X}, \tag{A.7}$$

where $\mathbf{1}_M$ is the $M \times 1$ all-ones vector $(11\ldots1)^{\mathrm{T}}$

The **v** vector's mean μ_v is

$$\mu_v = E\{\mathbf{v}\} = E\{\mathbf{u} - \mathbf{1}_M \otimes \mathbf{X}\} = -\mathbf{1}_M \otimes \mathbf{X} \tag{A.8}$$

and its covariance is

$$\begin{aligned}\mathbf{K}_v &= E\{(\mathbf{v}-E\{\mathbf{v}\})(\mathbf{v}-E\{\mathbf{v}\})^{\mathrm{T}}\}\\ &= E\{(\mathbf{u}-\mathbf{1}_M\otimes\mathbf{X}+\mathbf{1}_M\otimes\mathbf{X})\\ &\quad\times(\mathbf{u}-\mathbf{1}_M\otimes\mathbf{X}+\mathbf{1}_M\otimes\mathbf{X})^{\mathrm{T}}\}\\ &= E\{\mathbf{u}\mathbf{u}^{\mathrm{T}}\} = \mathbf{K}_u = \mathbf{I}_M\otimes\mathbf{K}_x.\end{aligned} \tag{A.9}$$

Therefore, **v** has a conditional probability density function (p.d.f) [18] of the form

$$p_{v|X}(\mathbf{V}) = \frac{1}{(2\pi)^{MN/2}|\mathbf{K}_v|^{1/2}}\mathrm{e}^{-(1/2)(\mathbf{V}-\mu_v)^{\mathrm{T}}\mathbf{K}_v^{-1}(\mathbf{V}-\mu_v)}, \tag{A.10}$$

where μ_v is a function of **X** as given by Eq. (A.8)

The difference vectors $\mathbf{r}^i$ defined in Eq. (A.5) have norm given by

$$\|\mathbf{r}^i\|^2 = (\mathbf{r}^i)^{\mathrm{T}}\mathbf{r}^i = \mathbf{v}^{\mathrm{T}}(\phi_M^i \otimes \mathbf{I}_N)\mathbf{v}, \tag{A.11}$$

where ϕ_M^i is the $M \times M$ *indicator matrix* where all elements are zero except the i-row-i-column element which is one.

The difference norm vector **z** is defined as

$$\mathbf{z} = (\|\mathbf{r}^1\|^2\|\mathbf{r}^2\|^2\ldots\|\mathbf{r}^M\|^2)^{\mathrm{T}} \tag{A.12}$$

and its characteristic function [18] is

$$\begin{aligned}M_z(\mathbf{s}) &= E\{\mathrm{e}^{-\mathbf{s}^{\mathrm{T}}\mathbf{z}}\} = E\{\mathrm{e}^{-\sum_{i=1}^M s_i\mathbf{v}^{\mathrm{T}}(\phi_M^i\otimes\mathbf{I}_N)\mathbf{v}}\}\\ &= \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}\mathrm{e}^{-\sum_{i=1}^M s_i\mathbf{V}^{\mathrm{T}}(\phi_M^i\otimes\mathbf{I}_N)\mathbf{V}}\\ &\qquad\times p_{v|X}(\mathbf{V})\,\mathrm{d}\mathbf{V}\\ &= \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}\mathrm{e}^{-\sum_{i=1}^M s_i\mathbf{V}^{\mathrm{T}}(\phi_M^i\otimes\mathbf{I}_N)\mathbf{V}}\\ &\quad\times\frac{1}{(2\pi)^{MN/2}|\mathbf{K}_v|^{1/2}}\mathrm{e}^{-(1/2)(\mathbf{V}-\mu_V)^{\mathrm{T}}\mathbf{K}_v^{-1}(\mathbf{V}-\mu_v)}\mathrm{d}\mathbf{V}\\ &= \left(\frac{|\mathbf{K'}_v|}{|\mathbf{K}_v|}\right)^{1/2}\mathrm{e}^{-(1/2)u_v^{\mathrm{T}}\mathrm{D}\mu_v}\int_{-\infty}^{\infty}\cdots\\ &\int_{-\infty}^{\infty}\frac{1}{(2\pi)^{MN/2}|\mathbf{K'}_v|^{1/2}}\mathrm{e}^{-(1/2)(\mathbf{V}-\mu'_v)^{\mathrm{T}}\mathbf{K'}_v^{-1}(\mathbf{V}-\mu'_v)}\mathrm{d}\mathbf{V}\\ &= \left(\frac{|\mathbf{K'}_v|}{|\mathbf{K}_v|}\right)^{1/2}\mathrm{e}^{-(1/2)u_v^{\mathrm{T}}\mathrm{D}\mu_v},\end{aligned}$$

where

$$\mathbf{K}'_v = \left(\mathbf{K}_v^{-1} + 2\sum_{i=1}^{M} s_i(\phi_M^i \otimes \mathbf{I}_N)\right)^{-1}$$

$$\mu'_v = \left(\mathbf{I}_{MN} + 2\sum_{i=1}^{M} s_i(\phi_M^i \otimes \mathbf{I}_N)\mathbf{K}_v\right)^{-1} \mu_v$$

$$\mathbf{D} = \left(\mathbf{K}_v + \sum_{i=1}^{M} \frac{1}{2s_i}(\phi_M^i \otimes \mathbf{I}_N)\right)^{-1}$$

and finally

$$M_z(\mathrm{s}) = \left(\left|\mathbf{I}_{MN} + 2\sum_{i=1}^{M} s_i\mathbf{K}_v(\phi_M^i \otimes \mathbf{I}_N)\right|\right)^{-1/2}$$

$$\times \mathrm{e}^{-(1/2)\mu_v^{\mathrm{T}}(\mathbf{K}_v+\sum_{i=1}^{M}(1/2)s_i(\phi_M^i\otimes\mathbf{I}_N))^{-1}\mu_v}. \quad \text{(A.13)}$$

Using Eqs. (A.9) and (A.8) and $\mathbf{I}_M = \sum_{i=1}^{M} \phi_M^i$ in (A.13) we get

$$M_z(\mathbf{s}) = \left|\sum_{i=1}^{M}\phi_M^i\otimes\mathbf{I}_N + 2\sum_{i=1}^{M}s_i(\mathbf{I}_M\otimes\mathbf{K}_x)(\phi_M^i\otimes\mathbf{I}_N)\right|^{-1/2}$$

$$\times \mathrm{e}^{-(1/2)(\mathbf{1}_M^{\mathrm{T}}\otimes\mathbf{x}^{\mathrm{T}})(\mathbf{I}_M\otimes\mathbf{K}_x+\sum_{i=1}^{M}(1/2)s_i(\phi_M^i\otimes\mathbf{I}_N))^{-1}(\mathbf{1}_M\otimes\mathbf{x})}$$

$$= \left|\sum_{i=1}^{M}(\phi_M^i \otimes \mathbf{I}_N + 2s_i\phi_M^i \otimes \mathbf{K}_x)\right|^{-1/2}$$

$$\times \mathrm{e}^{-(1/2)(\mathbf{1}_M^{\mathrm{T}}\otimes\mathbf{X}^{\mathrm{T}})(\sum_{i=1}^{M}((1/2)s_i\phi_M^i\otimes\mathbf{I}_N+\phi_M^i\otimes\mathbf{K}_x))^{-1}(\mathbf{1}_M\otimes\mathbf{x})}$$

$$= \left|\sum_{i=1}^{M}\phi_M^i \otimes (\mathbf{I}_N + 2s_i\mathbf{K}_x)\right|^{-1/2}$$

$$\times \mathrm{e}^{-(1/2)(\mathbf{1}_M^{\mathrm{T}}\otimes\mathbf{X}^{\mathrm{T}})(\sum_{i=1}^{M}\phi_M^i\otimes((1/2)s_i\mathbf{I}_N+\mathbf{K}_x))^{-1}(\mathbf{1}_M\otimes\mathbf{x})}$$

$$= \prod_{i=1}^{M}|\mathbf{I}_N + 2s_i\mathbf{K}_x|^{-(1/2)}\mathrm{e}^{-s_i\mathbf{X}^{\mathrm{T}}(\mathbf{I}_N+2s_i\mathbf{K}_x)^{-1}\mathbf{X}}$$

$$= \prod_{i=1}^{M} M_{z_i}(s_i). \quad \text{(A.14)}$$

Eq. (A.14) shows that, for fixed $\mathbf{X}$, the components $z_i = \|\mathbf{r}^i\|^2$ of the vector $\mathbf{z}$ are independent and identically distributed (iid) random variables. The cumulative distribution function of z_i is given by the inverse Laplace transform as

$$F_{z_i|\mathbf{X}}(Z_i,\mathbf{X}) = \int_0^{Z_i} p_{z_i|\mathbf{X}}(\lambda,\mathbf{X})\,\mathrm{d}\lambda$$

$$= \mathscr{L}^{-1}\left\{\frac{\mathbf{1}}{s}M_{z_i}(s)\right\}. \quad \text{(A.15)}$$

So, we can write

$$\Pr\{\|\mathbf{x}-\mathbf{y}^i\|^2 > Nd^*|\mathbf{X}\} = \Pr\{z_i > Nd^*|\mathbf{X}\}$$

$$= 1 - F_{z_i|\mathbf{X}}(Nd^*,\mathbf{X}) \quad \text{(A.16)}$$

and the conditional matching probability is

$$P_{m|\mathbf{X}}(N,M,d^*) = 1-(1-F_{z_i|\mathbf{X}}(Nd^*,\mathbf{X}))^M. \quad \text{(A.17)}$$

For finite M, there is a non-zero probability that a random dictionary does not include the optimum solution to the compression problem at rate zero, in this case the all-zeros vector. This random dictionary resembles the dictionary that MMP builds when the input vector is memoryless Gaussian. However, in MMP the initial dictionary includes the all-zeros vector (in fact, it includes all equal-components vectors). We then replace one of the dictionary vectors by the all-zeros vector $\mathbf{0}_N$, and evaluate the matching probability for the new dictionary with the zero included.

With vector $\mathbf{0_N}$ in the dictionary, Eq. (A.17) changes to

$$P_{m|\mathbf{X}}(N,M,d^*,\mathbf{X})$$

$$= \begin{cases} 1-(1-F_{z_1|\mathbf{X}}(Nd^*,\mathbf{X}))^{M-1}, & \|X\|^2 \\ \quad > Nd^*, & \\ 1, & \|X\|^2 \leqslant Nd^*. \end{cases}$$

$$= 1-(1-F_{z_1|X}(Nd^*,\mathbf{X}))^{M-1}S(\mathbf{X}^{\mathrm{T}}\mathbf{X}-Nd^*), \quad \text{(A.18)}$$

where

$$S(x)=\begin{cases}0, & x<0,\\ 1, & x\geqslant 0\end{cases}$$

is the unit step function.

The matching probability will be

$$P_m(N,M,d^*) = \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} P_{m|\mathbf{X}}(N,M,d^*,\mathbf{X})\,p_x(\mathbf{X})\,\mathrm{d}\mathbf{X}. \tag{A.19}$$

If the components of the Gaussian vector $\mathbf{x}$ are iid, its covariance matrix is $\mathbf{K}_x=\sigma_x^2\mathbf{I}_N$. Eq. (A.14) is then

$$M_{z_i}(s)=(1+2s\sigma_x^2)^{-N/2}\mathrm{e}^{-s\mathbf{X}^{\mathrm{T}}\mathbf{X}/(1+2s\sigma_x^2)}. \tag{A.20}$$

We then find the conditional matching probability using (A.15), (A.18) and (A.20) as

$$\begin{aligned}P_{m|\mathbf{X}^{\mathrm{T}}\mathbf{X}}&(N,M,d^*,\mathbf{X}^{\mathrm{T}}\mathbf{X})\\ &=1-\mathrm{e}^{-(M-1)(Nd^*+\mathbf{X}^{\mathrm{T}}\mathbf{X})/2\sigma_x^2}\\ &\times\left(\sum_{k=0}^{\infty}\frac{1}{k!}\left(\frac{\mathbf{X}^{\mathrm{T}}\mathbf{X}}{2\sigma_x^2}\right)^k\sum_{l=0}^{k+N/2-1}\frac{1}{l!}\left(\frac{Nd^*}{2\sigma_x^2}\right)^l\right)^{M-1}\\ &\times S(\mathbf{X}^{\mathrm{T}}\mathbf{X}-Nd^*).\end{aligned} \tag{A.21}$$

Eq. (A.21) shows that in the memoryless case $P_{m|\mathbf{X}}(N,M,d^*)$ is a function of the norm of $\mathbf{X}$ and is independent of the shape. So, we can define a matching probability conditioned to the norm as

$$\begin{aligned}P_{m|\,\|\mathbf{X}\|^2}&(N,M,d^*,\|\mathbf{X}\|^2)\\ &=1-\mathrm{e}^{-(M-1)(Nd^*+\|\mathbf{X}\|^2)/2\sigma_x^2}\\ &\times\left(\sum_{k=0}^{\infty}\frac{1}{k!}\left(\frac{\|\mathbf{X}\|^2}{2\sigma_x^2}\right)^k\sum_{l=0}^{k+N/2-1}\frac{1}{l!}\left(\frac{Nd^*}{2\sigma_x^2}\right)^l\right)^{M-1}\\ &\times S(\|\mathbf{X}\|^2-Nd^*).\end{aligned} \tag{A.22}$$

Since the conditional matching probability is independent of the shape, we can take the expected value using the p.d.f of the norm of $\mathbf{x}$ instead of the p.d.f of $\mathbf{x}$ when evaluating $P_m(N,M,d^*)$ in Eq. (A.19).

The components of the $\mathbf{x}$ vector are iid zero mean Gaussian variables with variance σ_x^2, so the sum of its squared components is *chi-square* distributed [18].

The matching probability is then

$$Pm(N,M,d^*) = \int_0^{\infty} P_{m|\lambda}(N,M,d^*,\lambda)\frac{\lambda^{N/2-1}\mathrm{e}^{-\lambda/2\sigma_x^2}}{(2\sigma_x^2)^{N/2}\Gamma(\frac{N}{2})}\,\mathrm{d}\lambda, \tag{A.23}$$

where $P_{m|\lambda}(N,M,d^*)$ is given by Eq. (A.22) with λ replacing $\|\mathbf{X}\|^2$.

Fig. 16 shows the matching probability given by Eq. (A.23) with $N=64,32,\ldots,2$ and $M=8192$ compared to experimental results. The experiment was carried on MatLab® with 1024 matching trials and averaging the results.

The probability distribution function $F_d(D)=\Pr\{d\leqslant D\}$ of the per-sample distortion d is given by Eq. (A.23) with $d^*=D$. So the conditional probability density function $f_{d|d\leqslant d^*}(D)$ of $d|d\leqslant d^*$, and the conditional expected value $E\{d|d\leqslant d^*\}$ are

$$f_{d|d\leqslant d^*}(D)=\frac{1}{F_d(d^*)}\frac{\mathrm{d}}{\mathrm{d}D}F_d(D)$$

$$\begin{aligned}E\{d|d\leqslant d^*\}&=\frac{1}{F_d(d^*)}\int_0^{d^*}\xi\,\frac{\mathrm{d}}{\mathrm{d}\xi}F_d(\xi)\,\mathrm{d}\xi\\ &=d^*-\frac{1}{F_d(d^*)}\int_0^{d^*}F_d(\xi)\,\mathrm{d}\xi.\end{aligned} \tag{A.24}$$

If there is a match, the distortion sum conditional expected value $E\{d|\text{match}\}$ for a vector of length N is given by Eq. (A.24)

$$\begin{aligned}Dm(N,M,d^*)&=E\{d|\text{match}\}\\ &=d^*-\frac{1}{F_d(d^*)}\int_0^{d^*}Pm(N,M,\xi)\,\mathrm{d}\xi.\end{aligned} \tag{A.25}$$

Fig 17 shows the distortion expected value in Eq. (A.25) with $N=64,32,\ldots,2$ and $M=8192$ compared to experimental results.

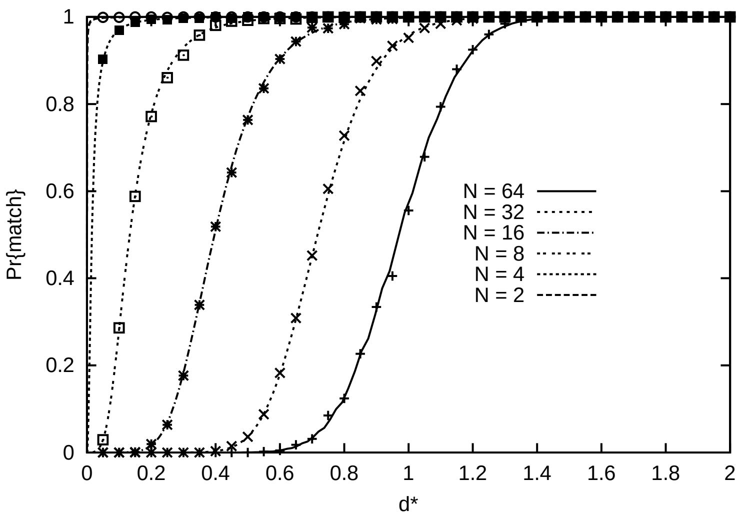


Fig. 16. Matching probability $Pm\times$ target distortion d^*, random dictionary with zero included.

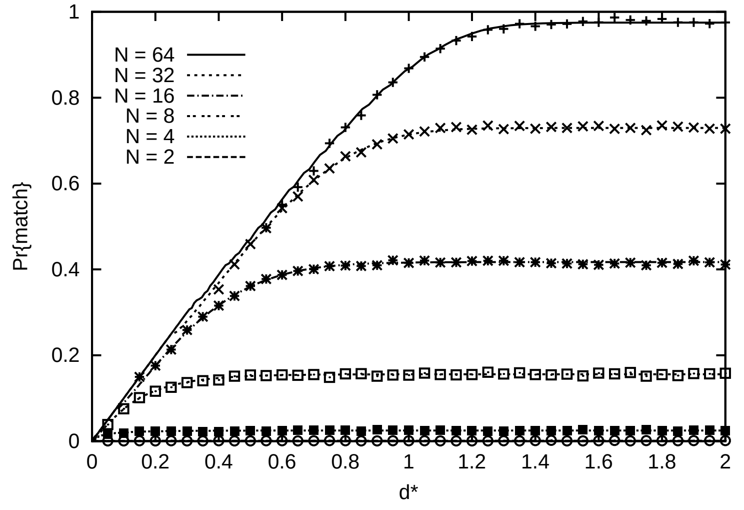


Fig. 17. $Dm\times$ target distortion d^*, random dictionary with zero included.

A.2. Matching Gaussian vectors at different scales

A distinctive feature of this work is the approximate matching of patterns using scales. Therefore, it is interesting to analyze the matching of a Gaussian vector of size N to vectors in a dictionary of Gaussian vectors of size $N/2, N/4, \ldots, 2$ interpolated to size N.

We will use the notation $\mathbf{y}^s = T^N_{N'}[\mathbf{y}]$ to denote the scaling operation that changes from a vector of size $N' \times 1$ to another vector of size $N \times 1$. One way to implement the interpolation operation is by use of a DCT [22]. If we want to change from a vector $\mathbf{y}$ of size $N' \times 1$ to a vector $\mathbf{y}^s$ of size $N \times 1 (N' \leqslant N)$, we first compute the N' point DCT of $\mathbf{y}$, given by $\mathscr{Y} = \mathbf{C}_{N'}\mathbf{y}$. Then we append $N - N'$ zeros to $\mathscr{Y}$ obtaining $\mathscr{Y}^s = \begin{pmatrix} \mathscr{Y} \\ \mathbf{0}_{N-N'} \end{pmatrix}$. Finally, we get the upsampled version by the N point inverse DCT of $\mathscr{Y}^s$, $\mathbf{y}^s = \mathbf{C}^T_N \mathscr{Y}^s$. Since the DCT is a unitary transformation, the matching probability

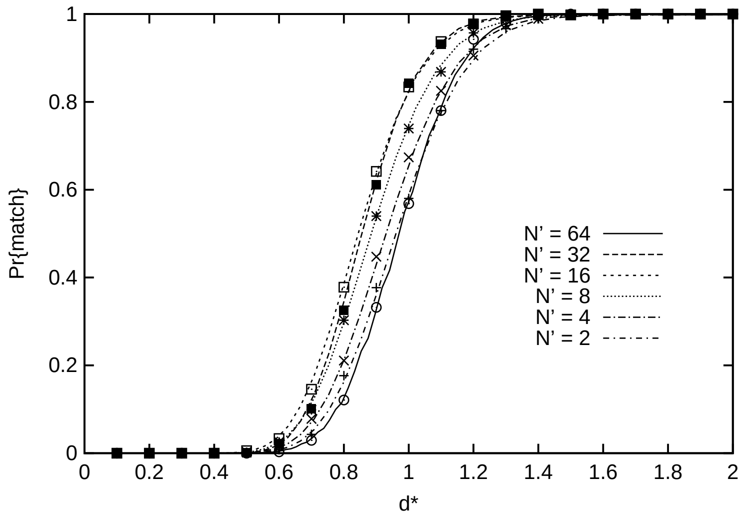


Fig. 18. Matching with scales for a dictionary with 8192 vectors.

can be evaluated in the transformed domain. In fact,

$$\begin{aligned}
&\Pr\{\|\mathscr{X}-\mathscr{Y}^s\|^2 \leqslant Nd^*\}\\
&=\Pr\{\|\mathbf{C}_N\mathbf{x}-\mathbf{C}_N\mathbf{y}^s\|^2 \leqslant Nd^*\}\\
&=\Pr\{(\mathbf{C}_N\mathbf{x}-\mathbf{C}_N\mathbf{y}^s)^{\mathrm{T}}(\mathbf{C}_N\mathbf{x}-\mathbf{C}_N\mathbf{y}^s) \leqslant Nd^*\}\\
&=\Pr\{(\mathbf{x}-\mathbf{y}^s)^{\mathrm{T}}\mathbf{C}_N^{\mathrm{T}}\mathbf{C}_N(\mathbf{x}-\mathbf{y}^s) \leqslant Nd^*\}\\
&=\Pr\{\|\mathbf{x}-\mathbf{y}^s\|^2 \leqslant Nd^*\}. \qquad \text{(A.26)}
\end{aligned}$$

Since the DCT is a linear transformation, the transformed version of the Gaussian vector $\mathbf{x}$ is also Gaussian [18] with covariance matrix given by

$$\mathbf{K}_{\mathscr{X}} = E\{\mathscr{X}\mathscr{X}^{\mathrm{T}}\} = E\{\mathbf{C}_N^{\mathrm{T}}\mathbf{x}\mathbf{x}^{\mathrm{T}}\mathbf{C}_N\} = \mathbf{C}_N^{\mathrm{T}}\mathrm{K}_{\mathbf{x}}\mathbf{C}_N. \qquad \text{(A.27)}$$

If $\mathbf{x}$ is memoryless then $\mathscr{X}$ will be also memoryless. In fact, $\mathbf{K}_{\mathbf{x}} = \sigma^2\mathbf{I}_N$ and $\mathbf{K}_{\mathscr{X}}$ is given by

$$\mathbf{K}_{\mathscr{X}} = \mathbf{C}_N^{\mathrm{T}}\sigma^2\mathbf{I}_N\mathbf{C}_N = \sigma^2\mathbf{C}_N^{\mathrm{T}}\mathbf{C}_N = \sigma^2\mathbf{I}_N = \mathbf{K}_{\mathbf{x}}. \qquad \text{(A.28)}$$

Therefore the matching probability of a $N \times 1$ memoryless Gaussian vector $\mathbf{x}$ considering a dictionary of scaled Gaussian vectors of original size $N' \times 1$ is the matching probability of an $N \times 1$ memoryless Gaussian vector with a dictionary of memoryless vectors of size $N' \times 1$ with $N - N'$ zeros appended to each of the dictionary vectors.

$$\begin{aligned}
&\Pr\{\|\mathbf{x}-T_{N'}^{N}[\mathbf{y}^i]\|^2 \leqslant Nd^*\}\\
&=\Pr\left\{\left\|\mathbf{x}-\begin{pmatrix}\mathbf{y}^i\\ \mathbf{0}_{N-N'}\end{pmatrix}\right\|^2 \leqslant Nd^*\right\}\\
&=\Pr\{\|\mathbf{x}'-\mathbf{y}^i\|^2+\|\mathbf{x}''\|^2 \leqslant Nd^*\}\\
&=E_{\mathbf{x}}''\{\Pr\{\|\mathbf{x}'-\mathbf{y}^i\|^2 \leqslant Nd^*-\|\mathbf{X}''\|^2\}\}, \qquad \text{(A.29)}
\end{aligned}$$

where $\mathbf{x}'$ is a memoryless Gaussian vector of size $N' \times 1$ and $\mathbf{X}''$ is a realization of the memoryless Gaussian vector $\mathbf{x}''$ of size $N - N' \times 1$. The matching probability with scales $Pm^s(N, N', M, d^*)$ is then

$$\begin{aligned}
&Pm^{\mathbf{s}}(N,N',M,d^*)\\
&=\int_0^{Nd^*} Pm\left(N',M,\frac{Nd^*-\lambda}{N'}\right)\\
&\quad\times\frac{\lambda^{(N-N')/2-1}\mathrm{e}^{-\lambda/2\sigma_x^2}}{(2\sigma_x^2)^{(N-N')/2}\Gamma\left(\frac{N-N'}{2}\right)}\,\mathrm{d}\lambda. \qquad \text{(A.30)}
\end{aligned}$$

Fig. 18 shows the matching probability with scales for a memoryless Gaussian vector of size 64×1 using

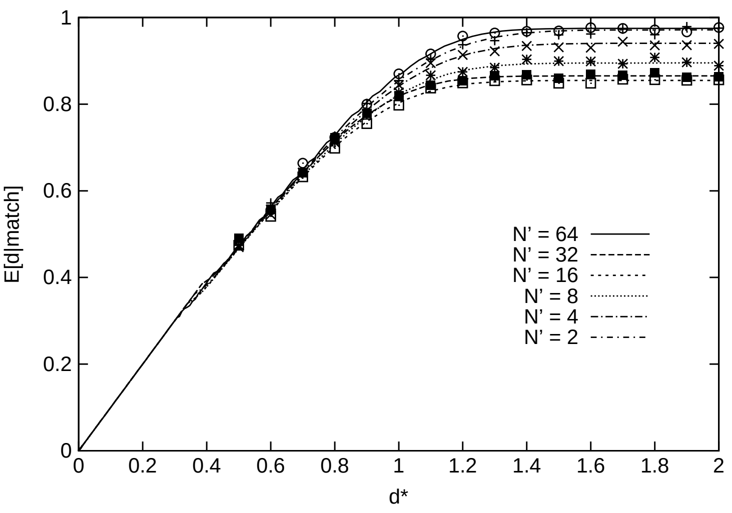


Fig. 19. Distortion in a match with scales for a dictionary with 8192 vectors.

a dictionary of 8192 vectors of length $64, 32, 16, \dots, 2$. It is interesting that, in this example, the probability of match for a dictionary of vectors using scales is greater than that of a dictionary of original size $N' = 64$. This can be understood considering:

- With $N = 64$ and $M = 8192$ the approximation of $\mathbf{X}$ using the dictionary without scales will have mean rate $R = 0.2031$ and mean distortion $D = 0.9747$. At this distortion level, a lowpass version of $\mathbf{X}$ will usually be sufficient to approximate the source. Therefore, there is little to gain with a dictionary of vectors of size $N' = 64$ over another dictionary of vectors of size $N' = 32$, for example.
- The matching with scales can be viewed as a match of vectors of size N' and a match of a vector of size $N - N'$ with the all-zeros vector. As the size N' decreases, it becomes easier to match the first part of size N' because we keep the dictionary size M fixed (this means we have more bits available to each component of the vector to match) but on the other hand it is harder to match the size-$(N - N')$ vector to the all-zeros vector. In the example illustrated in Fig. 18, the matching probability is maximum when $N' = 32$ or 16.

Fig. 19 shows the mean distortion given a match with scales for a memoryless Gaussian vector of size 64×1 using a dictionary of 8192 vectors of lengths $64, 32, 16, \dots, 2$. Since the matching probability with scales is greater in this case, the mean distortion is lower for the matching with scales.

We can conclude that *the matching with scales can improve the performance of approximate pattern matching at low rates.*

References

[1] M. Antonini, M. Barlaud, P. Mathieu, J. Daubechies, Image coding using wavelet transform, IEEE Trans. Image Process. 1 (April 1992) 205–221.

[2] M. Alzina, W. Szpankowski, A. Grama, 2D-pattern matching image and video compression: theory, algorithms, and experiments, IEEE Trans. Image Process. 11 (3) (March 2002) 318–331.

[3] 4.M. Atallah, Y. Genin, W. Szpankowski, Pattern matching image compression: algorithmic and empirical results IEEE Trans. Pattern Anal. Mach. Intell. 21 (1999) 618–627.

[4] R.A. Blahut, Principles and Practice of Information Theory, Addison-Wesley Publishing Company, Reading, MA, 1988.

[5] L. Bottou, P. Haffner, P.G. Howard, P. Sinard, Y. Bengio, Y. LeCun, High quality document image compression with DjVu, J. Electron. Imaging 7 (3) (1998) 410–428.

[6] M.B. de Carvalho, W.A. Finamore, Lossy Lempel–ziv on subband coding of images, IEEE International Symposium of Information Theory, June 27–July 1, Trondheim, Norway, 1994.

[7] M.B. de Carvalho, E.A.B. da Silva, A universal multi-dimensional lossy compression algorithm, 1999 IEEE International Conference on Image Processing, October 1999.

[8] M.B. de Carvalho, D.M. Lima, E.A.B. da Silva, W.A. Finamore, Universal multi-scale matching pursuits algorithm

with reduced blocking effect, 2000 IEEE International Conference on Image Processing, Vol. III, Vancouver, September 2000, pp. 853–856.

[9] M.B. de Carvalho, E.A.B. da Silva, W.A. Finamore, Rate distortion optimized adaptive multiscale vector quantization, 2001 IEEE International Conference on Image Processing, Thessaloniki, October 2001.

[10] C. Chan, M. Vetterli, Lossy compression of individual signals based on string matching and one pass codebook design, Proceedings of ICASSP'95, Detroit, 1995, pp. 2491–2494.

[11] C. Constantinescu, J.A. Storer, Improved techniques for single-pass adaptive vector quantization, Proc. IEEE. 82 (6) (June 1994) 933–939.

[12] M. Effros, P.A. Chou, R.M. Gray, One-pass adaptive universal vector quantization, Proceedings of ICASSP'94, Vol. 5, Adelaide, 1994, pp. 625–628.

[13] V.K. Goyal, M. Vetterli, Quantized overcomplete expansions in $\mathbb{R}^{\mathbf{N}}$: analysis, synthesis, and algorithms IEEE Trans. Inform. Theory 44 (1) (January 1998) 16–31.

[14] ISO/IEC JTCI/SC29/WGI (ITU/T SG28), JPEG2000 Verification Model 5.3, 1999.

[15] I. Kontoyiannis, An implementable Lossy version of the Lempel–Ziv algorithm—Part I: optimality for memoryless sources IEEE Trans. Inform. Theory 45 (1999) 2285–2292.

[16] T. Luczak, W. Szpankowski, A suboptimal Lossy data compression based in approximate pattern matching, IEEE Trans. Inform. Theory 43 (1997) 1439–1451.

[17] J.L. Mitchell, W.B. Pennebaker, C.E. Fogg, D.J. LeGall, MPEG Video Compression Standard, Chapman & Hall, New York, 1997.

[18] A. Papoulis, Probability, Random Variables and Stochastic Processes, McGraw-Hill Book Company, New York, 1984.

[19] W.B. Pennebaker, J.L. Mitchell, JPEG Still Image Compression Standard, Van Nostrand Reinhold, New York, 1993.

[20] A. Said, W.A. Pearlman, A new, fast and efficient image codec based on set partitioning in hierarchical trees, IEEE Trans. Circuits Systems Video Technol. 6 (June 1996) 243–250.

[21] G.J. Sullivan, R.L. Baker, Efficient quadtree coding of images and video, IEEE Trans. Image Process. 3 (3) (May 1994) 327–331.

[22] K.H. Tan, M. Ghanbari, Layered image coding using the DCT pyramid, IEEE Trans. Image Process. 4 (4) (April 1995) 512–516.

[23] P.P. Vaidyanathan, Multirate Systems and Filter Banks, Prentice-Hall Inc., Englewood Cliffs, NJ, 1993.

[24] E.H. Yang, J. Kieffer, On the performance of data compression algorithms based upon string matching, IEEE Trans. Inform. Theory 44 (1998) 47–65.

[25] J. Ziv, A. Lempel, Compression of individual sequences via variable-rate coding, IEEE Trans. Inform. Theory it-24 (5) (September 1978) 530–536.