UNIVERSAL MULTI-SCALE MATCHING PURSUITS ALGORITHM WITH REDUCED BLOCKING EFFECT

Murilo B. de Carvalho

Depto. de Eng. de Telecomunicações
Universidade Federal Fluminense
R. Passos da Pátria, 156
Niteroi - RJ
24210-240, BRASIL
murilo@lps.ufrj.br

Eduardo A. B. da Silva

PEE/COPPE/DEL/EE
Universidade Federal do Rio de Janeiro
Cx. P. 68504,
Rio de Janeiro, RJ
21945-970, BRASIL
eduardo@lps.ufrj.br

Weiler Alves Finamore

CETUC
PUC-Rio
Rio de Janeiro, RJ
BRASIL
weiler@cetuc.puc-rio.br

Denise de Melo Lima

PEE/COPPE
Universidade Federal do Rio de Janeiro
Cx. P. 68504, Rio de Janeiro, RJ
21945-970, BRASIL
dml@lps.ufrj.br

ABSTRACT

A method that reduces the blocking effect of the lossy compression of multi-dimensional data with a version of the UMMP (Universal Multi-scale Matching Pursuit) algorithm is presented. The method consists of a generalization of the previously presented UMMP [1] which extends its use to bases with overlapping functions. We circumvent the difficult optimization problem posed by overlapping functions at the encoder by the use of two different bases, one for the encoding and another, with overlapping functions, for the decoding of the data.

1. INTRODUCTION

In an earlier work [1], we described a new class of universal multi-dimensional lossy data compression algorithm, the UMMP (Universal Multiscale Matching Pursuits). UMMP is a compression scheme that decomposes the input signal using a dictionary of functions, in a similar way to the Matching Pursuits algorithm [2]. Differently from Matching Pursuits however, it builds its own dictionary as it encodes the data, instead of using a dictionary $\mathcal{D}$ which is fixed a priory. Also, the vectors in the dictionary can be *contracted* and *dilated* by the use of a scale transformation. To build the dictionary, UMMP relies on a segmentation procedure resembling that used by the lossless hierarchical procedure described on [3]. UMMP repeatedly parses a residue signal in smaller segments until it can be represented, within a given accuracy level, by the current dictionary elements. The dictionary is updated as the segments are joined to reconstruct the original signal. In this paper, we describe a version of the UMMP algorithm that has good performance for a large class of sources. In figure 4 we can see the performance of this version when applied to the image LENA 256×256 compared to the performance of the image-specific JPEG. The results are very good for a universal algorithm.

The one-dimensional UMMP operates on a vector $\mathbf{X} = (X_0 \dots X_{N-1})$. We assume that $N = 2^k$, k an integer, but it's easy to generalize the algorithm for cases other than these. UMMP uses a dictionary $\mathcal{D}$ of vectors, initially set to $\mathcal{D}_0 = \{\mathbf{s}_0, \dots, \mathbf{s}_{I-1}\}$ and a scale transformation $T_N^M : \mathbb{R}^M \to \mathbb{R}^N$ that maps a vector of size M into a vector of size N. The vectors in $\mathcal{D}$ can have different sizes. UMMP starts by scaling all the vectors in $\mathcal{D}_0$ to size N using a scale transformation $\mathbf{s}_i^{(N)} = T_N^{\ell(\mathbf{s}_i)}[\mathbf{s}_i]$, where $\ell(\mathbf{s}_i)$ is the length of $\mathbf{s}_i$. In the remaining of the paper we will drop the superscript $\ell(\mathbf{s}_i)$ and write just $T_N[\mathbf{s}_i]$ for the scale transformation. Next, one searches in the dictionary for the vector $\mathbf{s}_{i_{opt}}^{(N)}$ which minimizes the squared error $\xi = \|\mathbf{X} - \mathbf{s}_i^{(N)}\|^2$. If the squared error ξ is not greater than a given target distortion d^* times the vector size N, then the expansion is finished. Otherwise, the input vector $\mathbf{X}$ is split in two segments $\mathbf{X}' = (X_0 \dots X_{N/2-1})$ and $\mathbf{X}'' = (X_{N/2} \dots X_{N-1})$. Then

0-7803-6297-7/00/$10.00

all the vectors in $\mathcal{D}$ are scaled to size $N/2$ using a transformation $\mathbf{s}_i^{(N/2)} = T_{N/2}[\mathbf{s}_i]$. After that, the same procedure is recursively applied to $\mathbf{X}'$ and $\mathbf{X}''$. After returning from the recursive calls, we will have two approximations $\hat{\mathbf{X}}'$ and $\hat{\mathbf{X}}''$. We then form an estimate of the original input vector $\mathbf{X}$ by the concatenation of the segments as $\hat{\mathbf{X}} = (\ \hat{\mathbf{X}}' \quad \hat{\mathbf{X}}''\ )$. The dictionary $\mathcal{D}$ is updated by including $\hat{\mathbf{X}}$ in it.

2. BLOCKING EFFECT IN UMMP

UMMP attempts to find an approximate match to the input vector $\mathbf{X}$ using scaled versions of the vectors in the dictionary. It uses the function $T_N[\cdot]$ to match vectors of different sizes. If a match is not found, UMMP splits the input vector $\mathbf{X}$ in two segments $\mathbf{X}'$ and $\mathbf{X}''$. After finding approximations for both segments $\hat{\mathbf{X}}'$ and $\hat{\mathbf{X}}''$, the algorithm forms an estimate of the original vector by the concatenation of $\hat{\mathbf{X}}'$ and $\hat{\mathbf{X}}''$. This procedure guarantees that the mean distortion in each segment, and therefore in the concatenation, is less than or equal to the target distortion d^*. However it has no control regarding the continuity at the boundary of the two segments, unless the target distortion is set to zero. In fact, the approximation can be viewed as a sum of the two vectors $\hat{\mathbf{X}}'_s = (\ X'_0 \ \ldots \ X'_{N/2-1} \ \ 0 \ \ldots \ \ 0\ )$ and $\hat{\mathbf{X}}''_s = (\ 0 \ \ldots \ \ 0 \ \ X''_0 \ \ldots \ X''_{N/2-1}\ )$ that represent two non-overlapping functions. If the target distortion d^* is set to a smaller value, the algorithm will recursively split the input vector in more than two segments, but the approximation will always be a sum of non-overlapping functions. If we want the approximation to be smooth at the concatenation point for all $d^* > 0$, we can use overlapping functions instead. One way to achieve this is to change the scaling transformation $T_N[\cdot]$ to allow non zero values outside a given block. However, when using overlapped functions, we can't be sure that the two segments $\hat{\mathbf{X}}'$ and $\hat{\mathbf{X}}''$, each one with mean distortion below the target distortion, will sum to a block $\hat{\mathbf{X}}$ with mean distortion below d^*. Therefore the choice of the optimum dictionary vector to be used depends on the choice of vectors from the neighboring blocks.

To avoid the joint optimization problem we can use one basis, with non-overlapped functions, to analyze the data at the encoder side and a different basis, with overlapped functions, to synthesize the data at the decoder side. For example, lets consider that UMMP parses the input vector $\mathbf{X}$ in four segments of same length and approximates each one by an equal components vector. 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 downsampled by $N/4$ version of it. It's well known from filter banks theory [4] that we can recover a low-pass version of $\mathbf{X}$ from its downsampled version using a synthesis filter that is different from the analysis filter. So we can tailor the impulse response of the synthesis filter, or, on the UMMP decoder, the corresponding basis vector, to match the smoothness requirement.

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

- Firstly, we replace each $\mathbf{s}_{i_k}^{(N_k)}$ by a concatenation of vectors in the initial dictionary. That is, $\hat{\mathbf{X}} = \left(\ \mathbf{s}_{i_0}^{(N_0)} \quad \mathbf{s}_{i_1}^{(N_1)} \quad \ldots \quad \mathbf{s}_{i_{p-1}}^{(N_{p-1})}\ \right)$, that is, $\hat{\mathbf{X}}$ is composed by the concatenation of p segments, each one a version of a vector $\mathbf{s}_{i_k}$ in the dictionary scaled to size N_k.
- Next, we replace each $\mathbf{s}_{i_k}^{(N_k)}$ by a concatenation of vectors in the initial dictionary. That is, $\hat{\mathbf{X}} = \left(\ \mathbf{s}_{i_0}^{(L_0)} \quad \mathbf{s}_{i_1}^{(L_1)} \quad \ldots \quad \mathbf{s}_{i_{q-1}}^{(L_{q-1})}\ \right)$, where the vectors $\mathbf{s}_{i_k}$ are in the initial dictionary $\mathcal{D}_0$.
- Finally, we apply a zero-delay moving-average filter to $\hat{\mathbf{X}}$. The length of the filter at each component $\hat{X}_n$ of $\hat{\mathbf{X}} = (\ \hat{X}_0 \quad \ldots \quad \hat{X}_{N-1}\ )$ is proportional to L_k, the length of the segment $\mathbf{s}_{i_k}^{(L_k)}$ that contains the sample $\hat{\mathbf{X}}_n$. Therefore, the size of the filter is adjusted in a sample-by-sample basis. Considering again, for example, that the reconstructed $\hat{\mathbf{X}}$ has 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-overlapped flat vectors of size $N/4$ by four overlapped triangle-shaped vectors of size $N/2$.

Figure 1 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 is proportional to the size of the original reconstruction vector.

3. IMPLEMENTATION DETAILS

We have implemented UMMP in a computer program for simulation. The program was then used to lossy compress files containing gray-scale image data. In these experiments, we used an initial dictionary $\mathcal{D}_0 = \{-128, -124, \ldots, -4, 0, 4, \ldots, 124\}$ with 64 elements. The scale transformation $T_N[\cdot]$ used was a classical sampling rate change operation using a linear interpolator as the filter [4]. The images were divided in 8×8 blocks that were processed in sequence by the algorithm. The indexes output by the algorithm were encoded by an arithmetic coder. The two-dimensional UMMP used in the simulations is described next:

Let:

- $\mathbf{X}$ be an $N \times M$ matrix.
- $\mathcal{D}_0 = \{\mathbf{s}_0, \ldots, \mathbf{s}_{I-1}\}$ be an initial dictionary with I matrices of arbitrary dimensions.
- d^* be a target distortion.
- $T_{N,M}[\mathbf{X}]$ be a scale transformation that maps the matrix $\mathbf{X}$ in a matrix of size $N \times M$.

Procedure $\hat{\mathbf{X}} = \text{encode}(\mathbf{X}, \mathcal{D}, d^*)$:

step 1 scale all dictionary matrices to the size of $\mathbf{X}$, that is: $\mathbf{s}_i^{(N,M)} = T_{N,M}[\mathbf{s}_i]$ for all i.

step 2 find index i such that $||\mathbf{X} - \mathbf{s}_i^{(N,M)}||$ is minimum and make $\hat{\mathbf{X}} = \mathbf{s}_i^{(N,M)}$.

step 3 `if` $N = M = 1$, output index i and return $\hat{\mathbf{X}}$. `else` go to step 4.

step 4 `if` $||\mathbf{X} - \hat{\mathbf{X}}||^2 \leq NMd^*$ `then` output flag '1', output index i and return $\hat{\mathbf{X}}$. `else` go to step 5.

step 5 output flag '0'.

step 6 `if` $N > M$ `then` split $\mathbf{X} = \begin{pmatrix} \mathbf{X}' \\ \mathbf{X}'' \end{pmatrix}$, where $\mathbf{X}'$ and $\mathbf{X}''$ are $N/2 \times M$ `else` split $\mathbf{X} = \begin{pmatrix} \mathbf{X}' & \mathbf{X}'' \end{pmatrix}$ where $\mathbf{X}'$ and $\mathbf{X}''$ are $N \times M/2$

step 7 compute $\hat{\mathbf{X}}' = \text{encode}(\mathbf{X}', \mathcal{D}, d^*)$

step 8 compute $\hat{\mathbf{X}}'' = \text{encode}(\mathbf{X}'', \mathcal{D}, d^*)$

step 9 `if` $N > M$ `then` $\hat{\mathbf{X}} = \begin{pmatrix} \hat{\mathbf{X}}' \\ \hat{\mathbf{X}}'' \end{pmatrix}$, where $\hat{\mathbf{X}}'$ and $\hat{\mathbf{X}}''$ are $N/2 \times M$ `else` $\hat{\mathbf{X}} = \begin{pmatrix} \hat{\mathbf{X}}' & \hat{\mathbf{X}}'' \end{pmatrix}$ where $\hat{\mathbf{X}}'$ and $\hat{\mathbf{X}}''$ are $N \times M/2$

step 10 $\mathcal{D} = \mathcal{D} \cup \{\hat{\mathbf{X}}\}$

step 11 return $\hat{\mathbf{X}}$

4. EXPERIMENTAL RESULTS

Figure 2 shows the results of the original UMMP applied to the image LENA size 256 x 256. Figure 3 shows the results of the modified UMMP applied to the same image. Both images were coded at 0.40 bits/pixel. The effectiveness of the method proposed in reducing the blockiness is clear.

The PSNR values for the original and modified UMMP applied to LENA 256 x 256 at 0.40 bits/pixel are 28.7 and 28.5 dB, respectively. In figure 4 we show the objective performance of both decoders at other rates, and results for the JPEG algorithm for reference. The penalty in objective performance reaches a maximum of 0.66 dB at 1 bit/pixel, but the modified UMMP performs better than the original at low rates. In fact, at 0.14 bits/pixel its distortion is 0.36 dB below that of the original decoder. However, the subjective performance of the modified decoder is by far superior to that of the original decoder at all rates.

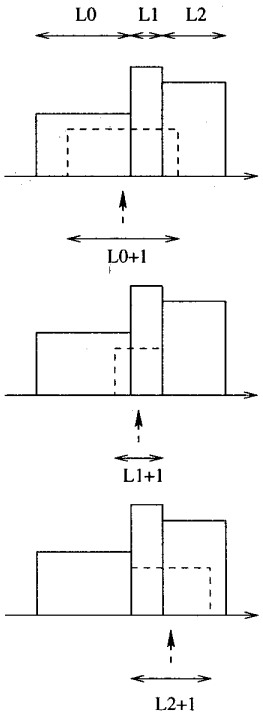

Fig. 1. FIR Post-filtering process.

5. CONCLUSION

We presented a blocking effect reduction method for a variant of the multi-dimensional algorithm for lossy data compression UMMP. It's clear from the simulation results that the method yields improvements in the subjective quality due to the reduction of the blocking effect. We avoided the need of joint optimization of neighboring blocks by use of different vectors in the dictionaries of the encoder and the decoder. We have made no attempts to optimize the two bases but we expect, however, to get further improvement by addressing that issue.

Fig. 2. Original UMMP.

Fig. 3. Modified UMMP.

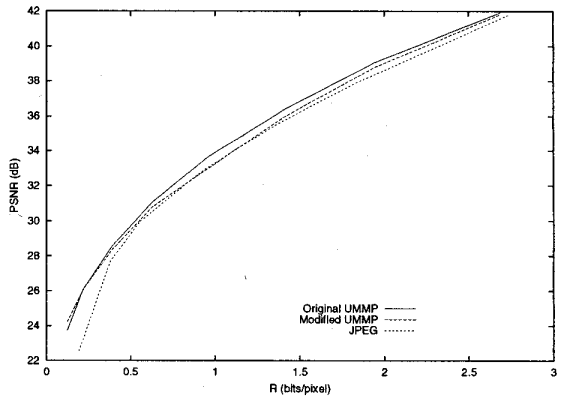

Fig. 4. Rate-distortion of UMMP and modified UMMP.

6. REFERENCES

[1] M. B. Carvalho and E. A. B. Silva, "A universal multi-dimensional lossy compression algorithm", *1999 IEEE International Conference on Image Processing*, October 1999, Kobe, Japan.

[2] S. G. Mallat and Z. Zhang, "Matching Pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, No. 12, pp. 3397-3415, December 1993.

[3] J. C. Kieffer, T. H. Park, Y. Xu, S. J. Yakowitz, "Progressive lossless image coding via self-referential partitions", *1998 IEEE International Conference on Image Processing*, October 1998, Chicago, Illinois.

[4] P. P. Vaidyanathan, "Multirate Systems and Filter Banks," Prentice-Hall Inc., 1993.