MULTISCALE (INTER/INTRA-FRAME) FRACTAL VIDEO CODING

Alexandru Bogdan

Department of Electrical Engineering and Center for Telecommunications Research
Columbia University, New York, NY 10027 USA
alex@ctr.columbia.edu
Proceedings of the IEEE Conf. ICIP-94, Nov. 1994, Austin, Tx.

ABSTRACT

In this work we present a *fractal image compression* algorithm for inter/intra-frame video coding. We have shown previously how the Iterated Transformation Theory (ITT) compression algorithm proposed by Jacquin can be modeled as the solution of a *second kind* two-scale functional equation. This approach allows us to introduce a *chain* of functional equations which we use to build a pyramid algorithm for still image compression. In this work we use the prediction property of the ITT-chain to create a inter-frame video coder. We combine the inter-frame video encoder with the still-image ITT-pyramid to generate a hierarchy of bit streams that can be used in multimedia applications.

1. INTRODUCTION

The coding method known as *fractal image compression* is based on the pioneering work of M. Barnsley [1] and, for automated image coding, on the ITT algorithm proposed by A. Jacquin [2, 3]. Properties such as high compression and natural image appearance at very low bit rates can make this algorithm a preferred choice for some image coding applications. ITT encoding exploits the self-similarity property found in images and natural phenomena in general. In video applications the algorithm has been used in two ways. Beaumont [4] has used a natural extension to 3-D blocks (space and time), which generates unpleasant artifacts. Other researchers have combined intra-frame fractal coding with classical video coding techniques like DCT and motion compensation [5].

We model Jacquin's ITT algorithm [2] as a linear two-scale functional equation in a Banach space (1). The ITT code is given by the equation and the reconstructed image is the unique solution f^*, if it exists. We can iterate this representation to generate a $ITT-chain$ (2) where the successive solutions of each equation are the frames in an ITT coded video sequence. Previously [6] we have used the $ITT-chain$ to generate a pyramidal multiscale still-image compression algorithm. We combine the inter-frame ITT video coder and the pyramid method into a double indexed chain which generates a hierarchy of bit streams that approximate the video signal at different resolutions.

This work was supported in part by the National Science Foundation under Grant CDR-88-11111.

2. FRACTAL IMAGE COMPRESSION

For a detailed description of the ITT algorithm we refer the reader to [2, 3]. Next we present some aspects relevant for the understanding of our method.

2.1. ITT fractal compression

An ITT coded image f is usually defined as the fixed-point of a contractive linear operator T defined on the finite-dimensional metric space $\mathcal{I}$ formed of all digital images with the same support. The operator $T = \sum_i T_i$ is the finite sum of piecewise continuous mappings T_i which have the property that the support of the range of T_i tile the image support $supp(f)$. In a discrete image representation, the range of T_i is a block of pixels we call the range block R_i, and the domain of T_i is the domain block D_i. Each T_i is parameterized with a few variables such that the description of T can be stored using fewer bits than the natural representation of the image f. The most used parameter space in ITT coding consists of the shape and position of the support of the ranges and domains of each map T_i, a multiplicative factor γ_i of the luminance levels, a luminance shift Δg_i, the choice of a symmetry operation which shuffles the pixels in a block, and a contraction factor of the support of the domain block. In plain words, each map T_i acts on a block of domain pixels D_i by filtering and subsampling, followed by a multiplication with γ_i, a shift in luminance by Δg_i and symmetry shuffling. The result replaces the

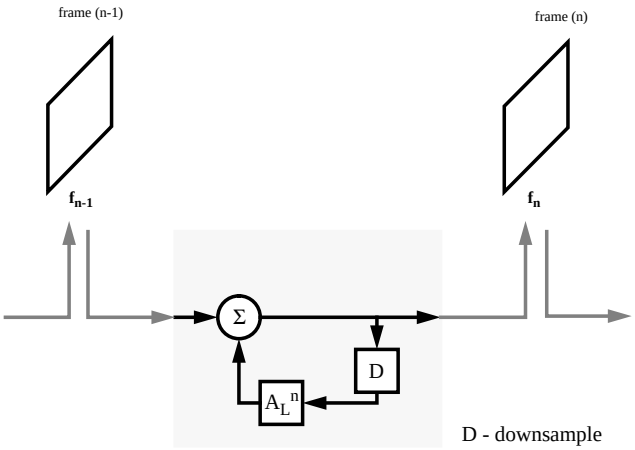

Figure 1: Decoding frame f_n from frame f_{n-1} and the ITT code A_L^n.

range block, which has fewer pixels and is usually at a position other than the domain block.

We can model the ITT algorithm as a linear functional equation [6] of the form

$$f(x) = Tf(x) = U_L f(x) + b = A_L f(2x) + b, \qquad (1)$$

in a Banach space where the image $f \in \mathcal{I}$ is a real function defined on a bounded rectangle in $\mathcal{R}^2$, x is the space coordinate and A_L is a linear operator. A discrete computer image representation of f makes the space $\mathcal{I}$ finite-dimensional, of size $\mathcal{R}^{mn}$ for images of size $m \times n$ pixels. The condition for the existence of a unique solution f^* for any $b \neq 0$, is that no eigenvalue of U_L is equal to 1. If all the eigenvalues are less than 1 in absolute value, then we have an iterative algorithm for solving (1) [7], of the kind used in ITT decoding. If an iterative solution is not possible, we can try a direct solution of the form $f = (I - U_L)^{-1} b$ using techniques from linear algebra [8]. In a finite-dimensional space setting, decoding the image f^* is equivalent to solving a large, sparse system of linear equations. Equation(1) is in the form of functional equations of the second kind [7, p363], which are known to be well behaved in general. It is remarkable that the parameterized ITT code is in fact a compact representation of the large sparse matrix U_L and that the iterative decoding process is a repeated use of a fast algorithm for the matrix vector multiplication $U_L f$.

2.2. The $ITT - chain$

In this section we show how we can define an $ITT - chain$ of functional equations when we iterate representation (1). The free term b groups the luminance shifts of each block which are defined by the parameter Δg_i in the original ITT algorithm [2]. Because b can be arbitrary and is also an element of the space of images $\mathcal{I}$, we can substitute any image for the free term in (1) and generate A_L through ITT encoding. In particular, if b is itself a fractal encoded image, we can iterate this process to obtain an ITT chain as

$$\begin{aligned} f_1(x) &= A_L^1 f_1(2x) + f_0(x) \\ f_2(x) &= A_L^2 f_2(2x) + f_1(x) \\ &\cdots \\ f_n(x) &= A_L^n f_n(2x) + f_{n-1}(x) \\ &\cdots \end{aligned} \qquad (2)$$

In Fig.1 we see how frame f_n is decoded using an iterative algorithm to solve equation n of the *chain* representation (2), when we know frame f_{n-1} and the linear operator A_L^n. In a direct decoding method, representations of the form (2) take the form $f_n = (I - U_L^n)^{-1} f_{n-1}$ where the linear operation of dilation by 2 symbolized by $f_n(2x)$ and the linear mapping A_L^n are concatenated into U_L^n.

2.2.1. Still image pyramid coding

Previously [6] we have used the ITT-chain to build a multiscale pyramid algorithm for still images, where we index the equations (2) by the scale factor. To generate a pyramid structure we iterate the functional equation at different scales. In order to have the same indexing as in (2), level "n" represents the original image f_n and at level 0 we have image f_0 of the lowest resolution, coded with large blocks of pixels. We associate the scale index with the size of the range blocks. In this case, f_n is tiled with square range blocks of size $2^0 = 1$ pixels, and at level k we tile the image f_{n-k} with square blocks with the side 2^k pixels. We start the encoding at the tip of the pyramid. Thus the code for f_0 results from tiling the original f_n with square range blocks of size 2^n pixels. Then we use the reconstructed f_0 and a tiling of the original image with range blocks of size 2^{n-1} to generate the linear operator A_L^1 in the first of the equations (2). At each level k the added detail is stored in the matrix A_L^k. To code f_0 we need f_{-1}, for which good choices are: an image with all gray levels constant at some value, a coarse approximation of f_n coded with a different algorithm, or we can take $f_{-1} = \mathbf{0}$ and solve an homogeneous equation.

In the last case f_0 is the eigenvector associated with the dominant eigenvalue of the linear operator U_L^0 and decoding is similar to solving an eigenvalue problem of linear algebra [9].

2.2.2. ITT inter-frame Video Coding

In the inter-frame video algorithm, we run the chain in the time direction (indexed by n). We start with frame f_0 which we can encode using the pyramid algorithm described in the previous subsection. Frame f_{n-1} is predicted by frame f_n which enters the functional equation as the affine term. We can use any fractal image coding method to generate variable tilings of the image such that a fidelity criterion is met for each frame. As an example in [2] *quad-tree* segmentation of a range block into four subblocks is performed if a suitable self-similar mapping T_i cannot be found. Then each subblock is encoded independently.

To illustrate the inter-frame predictive properties of the ITT-chain we rewrite the ITT coding of frame n from (2) as

$$(f_n - f_{n-1})(x) = A_L^n f_n(2x) = U_L^n f_n(x). \qquad (3)$$

We see from equation (3) that the linear operator U_L^n will map frame f_n on the difference $\Delta f_n = f_n - f_{n-1}$. Areas with no movement in the frame will generate areas of zero intensity in $\Delta f_n(x)$ such that U_L^n (or A_L^n) which is the ITT code for frame n codes only the changes between successive frames.

2.2.3. Hierarchical video coding

We can combine the pyramid and the inter-frame video coding methods, to generate a hierarchy of bit-streams, indexed by the time and scale variables. We can still use the spatial pyramid approach at each time index for intra-frame coding. A possible coding scheme is presented in Fig.2 where arrows show the direction of adding information through the linear mappings A_L. For clarity, the scale index which runs in the vertical direction, with the fine scale at the top, is not shown. We can define an hierarchy of bit streams starting with the base level which corresponds to the coarse encoding in the pyramid representation. We can use this coding structure in a multimedia video-conference application over a broadband communications network. Each receiver will extract from the total bit stream the number of levels it desires or is able to decode. A loss of information at levels other than the base level is localized at the individual frame and spatial location and will incur minimum damage. Thus it makes sense to have a higher level of protection against bit-errors

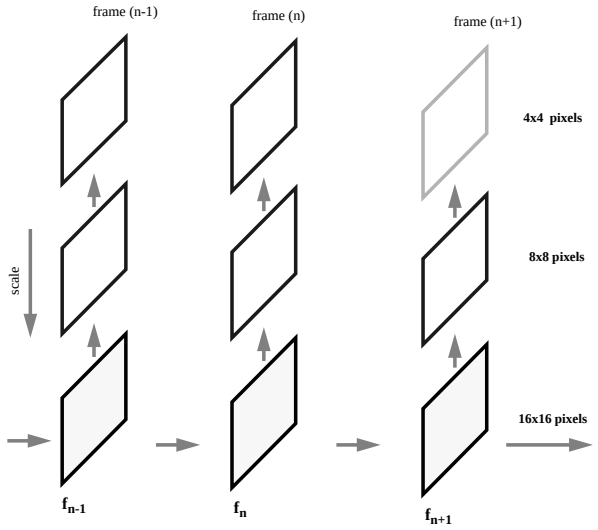

Figure 2: Pyramid fractal video coding with three levels. The base level is the low resolution, low bit rate based on 16x16 pixels blocks. Additional detail is presented in the higher resolution bit-streams and can be added depending on the quality of service required.

for the base level, and allow the high detail bit-streams to be transmitted at normal protection levels. Encoding can take place in the time and scale directions in parallel, with a time-delay between each level at the same frame number. Other bit-stream architectures are possible. An immediate extension is the coding of color video using the gray sequence as the prediction.

3. IMPLEMENTATION

In our software implementation, the low resolution base level (range blocks of 16x16 pixels) uses the inter-frame coding scheme while more levels of detail are added in the scale direction using range blocks of 8x8 and 4x4 pixels.

3.1. Encoding

At each stage, the current frame is tiled with square range blocks of the same size. For each range block we have to find a domain block twice the size (dilation by 2 in each spatial direction) somewhere in the frame, which we can map using subsampling, a multiplication by γ_i and a symmetry pixel shuffling to approximate well the range block. We classify all possible range and domain blocks in two classes: *flat* and *active*. More detailed encoding algorithms can be found in [3]. A block is considered *flat* if the variance in pixels intensities is

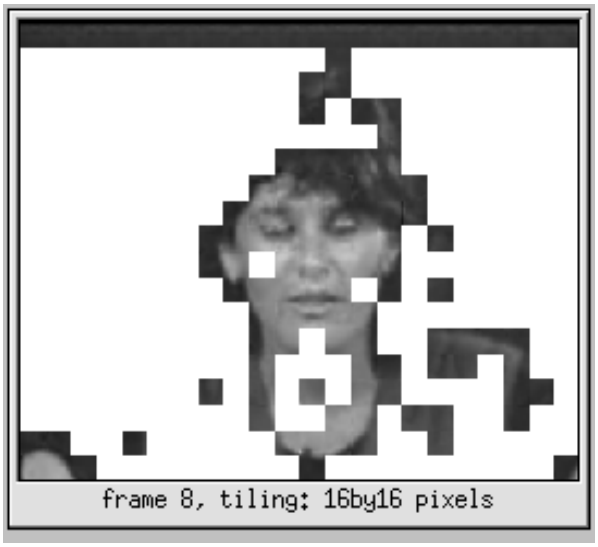

Figure 3: Frame 8, first level from the sequence "Miss America": range blocks (16x16 pixels) encoded by the ITT code A_L^8. Only 117 out of 396 range blocks are encoded. The white area is copied from frame 7.

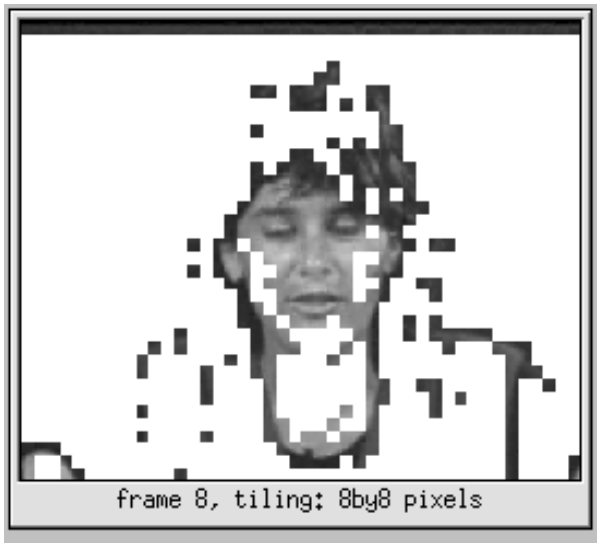

Figure 4: Frame 8, second level from the sequence "Miss America": range blocks (8x8 pixels) encoded by the ITT code A_L^8. Only 339 out of 1584 range blocks are encoded. The white area is copied from the first level.

below a certain threshold

$$\sum_k (f_k - \bar{f}_i)^2 \leq \epsilon_f, \quad i = 1, \ldots, N, \tag{4}$$

where N is the number of range blocks, ϵ_f is set empirically, $\bar{f}_i$ is the mean of block i and the summation is over all pixels in the block R_i or D_i. All other blocks are classified as *active*. All *flat* range blocks are not encoded ($\gamma_i = 0$) and we have to match the range and domain blocks labeled as *active*. Because the the number of domain blocks to be examined is quite low, we can use an exhaustive search algorithm in the encoder. Table 1 shows how we allocate the bits in the encoding of each *active* block.

flag	γ_i	domain address	symmetry	total
1	5	8+8=16	3	25 bits

Table 1: Bit allocation for an encoded range block R_i. The *flag* bit signals that the range block is encoded. A *flat* block is encoded using only the flag bit.

3.2. Decoding

In a digital coder, A_L is a very sparse matrix having a special structure generated by the well-known self-similar block matching encoding. While ITT encoding is computationally intensive, it was shown that decoder complexity is comparable with that of an adaptive DCT method [3]. When one of the range blocks is classified as *flat*, a block in the matrix representation of A_L is filled with zeros, resulting in additional savings in computation. Decoding at each stage is performed by iterating equation (1) a few times. Iterative methods usually use a stopping criterion such as setting a threshold not to be exceeded by the difference between two successive iterations. In our simulations we found that we can use a fixed number of iterations, 5 in this case, which give good numerical and visual results. An intuitive motivation is the fact that our mappings are usually strongly contractive because they map a full frame on the difference between two video frames.

3.3. Simulation

Simulation results for the sequence "Miss America" (352x288 pixels) coded into three bit streams (levels) are presented in Figs.5 and 6. Level 1 is the base level and uses 16×16 pixels range blocks for coarse approximation and compression in the range of $270 : 1$. Level 2 is coded with 8×8 blocks, and level 3 is the highest resolution level with 4×4 blocks and total compression of $\approx 26 : 1$. The high compression ratio is achieved twofold, by coding each block with a fixed number of bits as in Table 1, and by not coding the *flat* range blocks at each stage. Figure 3 shows that we encode (frame 8) only 117 of the 396 possible range blocks at the base level, the other 279 range blocks are copied from frame 7. At the second level

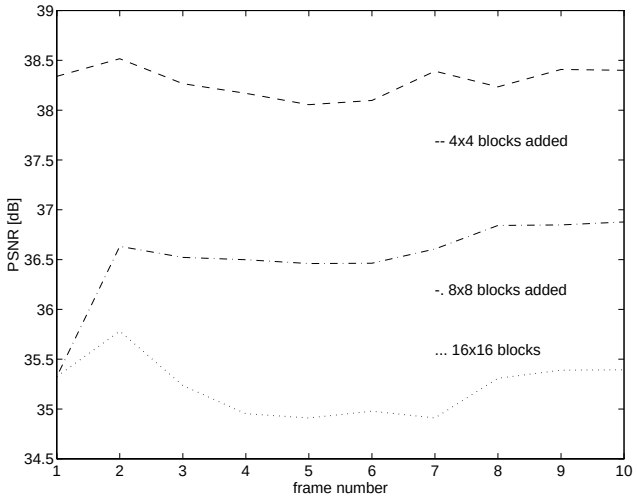

Figure 5: Coding results for the first ten frames of "Miss America": distortion evaluation.

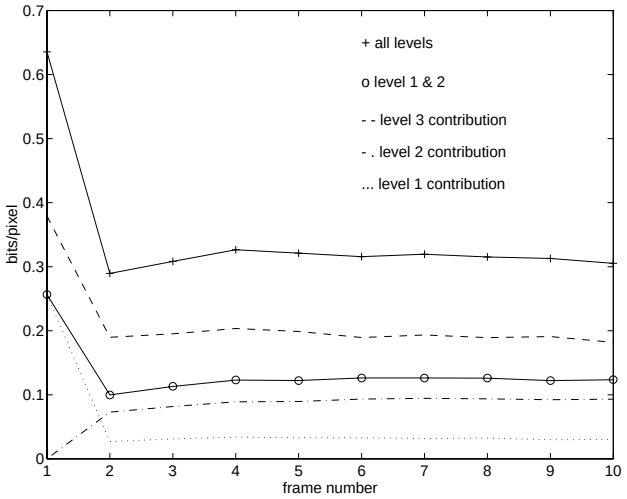

Figure 6: Coding results for the first ten frames of "Miss America": bit rates. Level 1: coding with blocks of 16x16pixels, level 2: 8x8 pixels, level 3: 4x4 pixels.

of the hierarchy (Fig.4) we have to encode 339 out of 1584 range blocks, the others being predicted from the base level. At the third level (4x4 pixels/block, not shown), predicted by the second level, we code only 685 of the possible 6336 range blocks. The first frame was encoded using the pyramid scheme (intra-frame) for still images. It is evident from the graphs that the the fidelity of the reconstruction, and the compression factors are substantially improved by the inter-frame fractal coding method.

4. CONCLUSIONS

We present in this work a method for inter/intra frame fractal video compression based on the fractal image compression algorithm and multiscale pyramid coding schemes. Our method is based on an $ITT - chain$ of functional equations which is obtained by iterating the well-known fractal image compression algorithm. When the $ITT-chain$ is iterated in the *time* direction, it generates an inter-frame video coding algorithm, while when the equations are indexed by *scale* we build a pyramid still image encoder. We combine the two schemes to obtain an hierarchy of bit-streams that can be used in multimedia applications. The quantitative evaluation of the algorithm is in the same range if not better than other published results for low-rate video coding [5].

5. REFERENCES

[1] M. F. Barnsley and L. Hurd, *Fractal Image Compression*. Wellesley, MA 02181: AK Peters, 1993.

[2] A. E. Jacquin, "Fractal image coding: a review," *Proceedings of the IEEE*, vol. 81, pp. 1451–1466, Oct. 1993.

[3] Y. Fisher, E. Jacobs, and R. Boss, "Fractal image compression using iterated transforms," in *Image and text compression* (J. A. Storer, ed.), ch. 2, pp. 35–61, Kluwer Academic, 1992.

[4] J. M. Beaumont, "Image data compression using data techniques," *BT Technol J*, vol. 9, pp. 93–109, Oct. 1991.

[5] B. Hürtgen and P. Büttgen, "Fractal approach to low-rate video coding," *SPIE Visual Communications and Image Processing*, vol. 2094, pp. 120–131, Nov. 1993.

[6] A. Bogdan, "Multiscale fractal image coding and the two-scale difference equation," Tech. Rep. CU/CTR/TR 358-94-05, Columbia University, Center for Telecommunications Resesarch, New York, NY 10027, Mar. 1994.

[7] L. Kantorovich and G. Akilov, *Functional Analysis*. New York: Pergamon Press, 2nd ed., 1982.

[8] W. Hackbusch, *Iterative solution of large sparse systems of equations*. No. 95 in Applied Mathematical Sciences, Springer-Verlag, 1994.

[9] Y. Saad, *Numerical methods for large eigenvalue problems*. Algorithms and architectures for advanced scientific computing, New York: Manchester University Press, 1992.