Fractal Block Coding of Digital Video

M. S. Lazar, *Student Member, IEEE*, and L. T. Bruton, *Fellow, IEEE*

***Abstract*—A video coding method is proposed which is based upon fractal block coding. The method utilizes a novel three-dimensional partitioning of input frames for which a number of efficient block-matching search methods can be used, and permits spatio-temporal splitting of the input blocks to improve overall-encoding quality. After describing the basic fractal block coding algorithm, the details of the proposed three-dimensional algorithm are presented along with encoding and decoding results from two standard video test sequences, representative of video-conferencing data. These results indicate that average compression rates ranging from 40 to 77 can be obtained with subjective reconstruction quality of video-conferencing quality. The results also indicate that, in order to meet the compression rates required for very low bit rate coding, it is necessary to employ additional techniques such as entropy encoding of the fractal transformation coefficients.**

I. Introduction

Fractal block coders (FBCs) have recently received considerable attention in the context of image coding [1]–[17]. FBC algorithms, which rely upon the theory of iterated contractive transformations [1], yield high compression ratios and can be used to provide simulated detail at resolutions higher than that of the original image [2]. Although the theory underlying the FBC algorithm is independent of the number of dimensions of the input signal, there are practical considerations which differ when encoding signals of different dimension. In this contribution, a three-dimensional (3-D) fractal block coding algorithm suitable for coding digital video signals is introduced. Unlike the intra-frame video coder presented in [3], the proposed method is based upon encoding three-dimensional (3-D) data blocks and is thus a true inter-frame coder.

One aspect of the 2-D FBC algorithm is the large associated encoding time, due predominately to an extensive matching search performed for each block of the input signal. For image encoding, several authors have presented methods to reduce encoding times [7]–[14]. In order to provide an efficient mechanism for 3-D block matching, a novel partitioning scheme of the input data frames is introduced for which several efficient search methods can be used. Specifically, target blocks of the matching process, called *range* blocks, are selected from *R-Frames*, which are consecutive, temporally non-overlapping groups of input frames. The source blocks for the matching process, called *domain blocks*, are selected from *D-Frames*, which are consecutive groups of possibly overlapping input frames.

This paper is organized as follows: in Section II a brief review of the basic fractal block coding algorithm and some of the associated theoretical background is presented; in Section III, the various extensions and methods used in applying the FBC algorithm to the coding of digital video signals is introduced; test results, computation of compression rates, computational considerations, and possible extensions to the proposed algorithm are presented in Section IV; finally, concluding remarks are given in Section V.

Manuscript received October 14, 1993; revised February 8, 1994. It was recommended by Kou-Hu Tzou.

The authors are with the Department of Electrical and Computer Engineering, The University of Calgary, Calgary, Alberta, Canada, T2N 1N4.

IEEE Log Number 9402111

II. Review of Fractal Block Coding

Much of the seminal work on fractal block coders has been published by A. Jacquin [5] and M. F. Barnsley, et. al., [1]–[4]. Although several variations and improvements have been subsequently proposed to their methods [7]–[14], most FBC type algorithms share similar characteristics. In this section, the basic FBC algorithm for m-dimensional (m-D) signals is described, and these common characteristics are highlighted. We begin by briefly reviewing some of the necessary theory from metric spaces which may be found, in greater detail, in [1], [2], [5].

A. Relevant Properties of Metric Spaces

Consider the complete metric space, $(\Im_m, d)$, where $\Im_m$ is the space of discrete domain (i.e. sampled) finitely bounded, real valued, m-D signals of size $N_1 \times N_2 \times \cdots \times N_m$, (where $N_1, N_2, \ldots, N_m \in Z, Z = \{0, 1, \cdots\}$), such that $\Im_m \subset (Z \times Z \times \cdots \times Z \times R)$, and d is the (normalized) Euclidean distance between the *sample* values of two members of $\Im$. For $a, b \in \Im_m$, d is defined by

$$d(a,b) = \frac{1}{N_1 N_2 \cdots N_m} \times \left(\sum_{i_1=1}^{N_1} \sum_{i_2=1}^{N_2} \cdots \sum_{i_m=1}^{N_m} (a(i) - b(i))^2 \right)^{1/2} \in R, \tag{1}$$

where $i = (i_1, i_2, \ldots, i_m)(1 \le i_k \le N_k, k = 1, 2, \ldots, m)$, and $a(i), b(i)$, represent the signal sample values at the location specified by the m-tuple $(i_1, i_2, \ldots, i_m)$.

For the space $\Im_m$, the mapping, $\Phi : \Im_m \rightarrow \Im_m$ is said to be contractive [1] if and only if

$$d(\Phi(a), \Phi(b)) \le s \cdot d(a,b), \quad 0 \le s < 1, \quad \forall (a,b) \in \Im_m. \tag{2}$$

If $\Im_m$ is a complete metric space, then Banach's fixed point theorem [1] ensures that each contractive mapping $\Im_m$ in has associated with it a unique fixed point (attractor), p_f, such that

$$\Phi(p_f) = p_f. \quad (3)$$

Moreover, if Φ is contractive then Φ may be iteratively applied to *any* point, $c \in \Im_m$, yielding, in the limit, the fixed point (also known as the attractor) of Φ[1]; that is,

$$\lim_{n\to\infty} \Phi^{\circ n}(c) = p_f, \quad \forall c \in \Im_m, \quad (4)$$

where "$\circ$" represents function composition.

B. Domain and Range Block Mapping

The underlying principle of fractal coding algorithms can be understood in terms of (4). Given an input signal, $q \in \Im_m$, the goal of the fractal coder is to find a contractive transformation, Φ, for which q is sufficiently close to the fixed point of Φ. If such a Φ can be found, then Φ can be used as the code for q. Decoding consists of starting with an arbitrary starting signal, $c \in \Im_m$, and iteratively applying Φ until the fixed point is reached.

Rather than attempt to find a contractive transformation which acts upon the entire signal q, as in the original IFS system proposed by Barnsley in [1], fractal block coders partition the input signal into range blocks, r_i, each of which gets encoded by a single transformation, τ_i. The source of each range block transformation is a domain block, d_j, taken from the same signal. This block-wise "self-transformability" results in the original signal being encoded in terms of itself; it is for this reason that the term "fractal" is used [5]. Some authors, however, prefer to think of this coding method as "attractor" based [10].

The non-overlapping partition of the input signal, q, called the range pool, $\{r_j\}$, is formed from the input signal as follows

$$q = \bigcup_{i=1}^{N_R} r_i, \quad r_i \cap r_j = \emptyset \text{ for } i \neq j, \quad r_j \in \Re_0, \quad (5)$$

where the size of each range block is given by $B_1 \times B_2 \times \cdots \times B_m$, $\Re_0$ represents the space of discrete range blocks of size $B_1 \times B_2 \times \cdots \times B_m$, and where r_i represents the restriction of the input signal to the i^{th} range block. The domain pool, $\{d_j\}$, which serves as the basis for coding the range blocks, is comprised of partitioning the input signal into a set of possibly overlapping blocks of size $D_1 \times D_2 \times \cdots \times D_m$,

$$q = \bigcup_{j=1}^{N_o} d_j, d_j \in \Delta_0, \quad (6)$$

where Δ_0 is the space of discrete domain blocks of size $D_1 \times D_2 \times \cdots \times D_m$, with (typically) $D_i \geq B_i$.

The mapping for the i^{th} range block, $\tau_i : \Delta_0 \to \Re_0$, consists of a contrast scaling, a_i, offset o_i, pixel[1] shuffling (isometry) $\imath_k(\bullet) : \Re_0 \to \Re_0$, and m-D "spatial" contraction, $S(\bullet) : \Delta_0 \to \Re_0$, operations [5]. The result of applying this mapping is an approximation to the i^{th} range block, $\tilde{r}_i$, which we write as follows:

$$\tilde{r}_i = \alpha_i \imath_{I(i)}(S(d_{N(i)})) + o_i, \quad (7)$$

where $N(i)$ is a domain block selection function, which associates the i^{th} range block with a domain block from $\{d_j\}$, and $I(i)$ is an isometry selection function, which maps the i^{th} range block to one of a set of possible isometry operations.

In addition to the range block mapping described by (7), several other functions and mappings have been proposed in the literature [10] [11] [14]. In this contribution, our attention is restricted to the range block mapping given in (7), which is based on Jacquin's work. Other proposed mappings can, in most cases, easily be accommodated within the video coding framework developed herein.

C. Domain Block Searching and Multi-Level Partitioning Within Fractal Block Coders

The FBC encoding process consists of determining, for all range blocks, the mapping parameters in (7) such that the distance between each range block and its approximation, $d(\tilde{r}_i, r_i)$, is minimized. The set of resulting transformations, τ_i, is then used as the resulting code for the signal. Entropy encoding can be used to further compress the transformations [2].

Typically, the Euclidean metric described in (1) is used for the distance function, $d(\tilde{r}_i, r_i)$, in which case there exists a closed form expression for evaluating α_i and o_i for a given $d_j, S(\bullet)$, and $\imath_k(\bullet)$ [15]. Since the spatial contraction operator is independent of the range block (it is fixed throughout encoding/decoding), the FBC encoding process requires that α_i and o_i be computed for candidate values of d_j and $\imath_k(\bullet)$, with those parameters resulting in the minimum approximation error selected for the final code, τ_i.

It is the search over possible d_j and $\imath_k(\bullet)$ which results in the large encoding times associated with the FBC algorithm. Several authors have proposed various approaches to reduce this search space. These methods include the use of a small image-independent set of domain blocks along with image dependent blocks [11], orthogonalizing the space spanned by the domain blocks [10], searching only domain blocks which originate from the area near the range block [12], the use of multiresolution information to constrain the domain block search and searching randomly selected areas of domain pool [7].

Similar to all block coding algorithms, FBCs also exhibit trade-offs between block size and the amount of compression achieved. Rather than use a fixed range block size, it has been proposed to use various block sizes depending upon the approximation error obtained per block [5] [12] [13] [15]. A simple and efficient method to incorporate different range block sizes when encoding images is the quadtree approach [15]. Extending the quadtree approach to m-D, a range block, originally of size $B_1 \times B_2 \times \cdots \times B_m$, is split into 2^m blocks each of size $B_1/2 \times B_2/2 \times \cdots \times B_m/2$ if the error, $d(\tilde{r}_i, r_i)$, is beyond a certain threshold.

[1]We take the liberty to use the term "pixel" to refer to the real-valued elements within each range and domain block, regardless of the dimension of the block.

If the space of discrete range blocks of size $B_1/2^r \times B_2/2^r \times \cdots \times B_m/2^r$ is denoted by $\Re_r$, then, for a maximum of N_q quad-tree partitions, the m-dimensional partitioning of the input signal given in (5) can be re-written as

$$q = \bigcup_{i=1}^{N_R} r_i, \quad r_i \cap r_j = \emptyset \text{ for } i \neq j, r_i \in \{\Re_0, \Re_1, \ldots, \Re_{N_q}\}. \tag{8}$$

Using this notation, the transform for each range block within a multi-level partitioned scheme is written as $\tau_i : \Delta_r \to \Re_r$, where Δ_r is the space of discrete domain blocks of size $D_1/2^r \times D_2/2^r \times \cdots \times D_m/2^r$. In this case, the contraction and isometry operations in (7) are generalized to $S(\bullet) : \Delta_r \to \Re_r$ and $\imath_k(\bullet) : \Re_r \to \Re_r, 0 \leq r \leq N_q$, respectively. Note that N_R is dependent upon the splitting error threshold level and partitioning levels, N_q, used. Thus, unless special restrictions are made, a multi-level partitioned FBC requires a variable rate coding scheme.

D. Contractivity Considerations

Given the transform for each range block, τ_i, we may write the transform for the entire signal, τ, as

$$\tau = \bigcup_{i=1}^{N_R} \tau_i, \quad \tau : \Im_m \to \Im_m. \tag{9}$$

If τ is contractive, then (4) can be used to perform the decoding operation. Specifically, the signal may be decoded by iteratively applying τ to any point, $f_0 \in \Im_m$, where the n^{th} iterate is given by

$$f_n = \tau^{\circ n}(f_0). \tag{10}$$

The fixed point of τ, f_p, will be the decoded signal. For the encoding of images ($m = 2$), the number of iterations required for convergence of (10) is small (we have found that, in practice, f_6 can be used as an approximation of f_p with little error).

It is important to understand the range of parameter values in (7) such that τ is sufficiently contractive for convergence of the decoding algorithm. Viewed strictly at a block level, it is necessary that $|\alpha_i| < 1$ in order for the corresponding τ_i to be contractive (using the Euclidean metric). In practice, however, this condition has been found to be unnecessarily restrictive [8] [16] [17]. This is due to the fact that each τ_i is dependent upon other τ_i which have been applied at previous decoding iterations. Thus, even if there are some τ_i for which $|\alpha_i| > 1$, their behaviour may be dominated by other τ_i which are sufficiently contractive to allow the entire signal transform, τ, to be contractive and hence for decoding to converge. A transformation, τ, for which this is case is known as an eventually contractive mapping [10], [17], [18].

It is also interesting to note that, in order to be contractive under the Euclidean distance metric, it is not strictly necessary that (7) contain the m-D spatial contraction operator, $S(\bullet) : \Delta_r \to \Re_r$. This is a result of the metric (1) being defined only over pixel values and not upon any "spatial" properties of the domain block. It is, therefore, *possible* that the domain blocks and range blocks be of the same size. *The principal motivation for using the operator, $S(\bullet)$, however, is to propagate signal detail from one scale to another.* A more detailed discussion of this issue can be found in [15].

III. Three Dimensional Fractal Block Coding

In this section, a three-dimensional (3-D) FBC algorithm suitable for the encoding and decoding of digital video signals is described. Rather than apply the 2-D FBC algorithm on an intra-frame basis [3], the proposed coder uses 3-D range and domain blocks. *The use of 3-D blocks has the potential for higher compression rates than intra-frame coding since the number of additional coefficients required to encode each range block can be selected to be considerably smaller than the pixels within the extra frames included in the range block.*

In order to simplify the number of possible 3-D operations, we adopt the philosophy that spatial and temporal operations be distinct. In other words, operations on domain and range blocks are applied first to the pixels within each frame of the block, then upon the frames themselves.

A. Partitioning the Input Signal into R-Frames and D-Frames

One of the principal differences between 2-D and 3-D FBC coding is the fact that 2-D signals (images) are naturally bounded in all dimensions, whereas video signals are naturally bounded in only two (spatial) dimensions and can be considered, for practical purposes, infinite in the third (temporal) dimension. It is therefore important that any video block coding scheme contain a method of temporal partitioning. For example, in the recently proposed MPEG-1 video coding scheme, it has been suggested that, for a large class of video sequences, the difference between interpolated frames (I-Frames) (from which predicted frames are derived) should be about 10 frames [19].

For the proposed 3-D FBC algorithm, we permit the original size of range blocks and domain blocks to be $B \times B \times T$ and $M_1 B \times M_2 B \times M_3 T$, respectively, where M_1, M_2 are spatial scaling values, and M_3 is a temporal scaling value. Range blocks are selected from *R-Frames*, which are consecutive, non-overlapping groups of input frames. The length of each *R-Frame* is restricted so that an integer number of range blocks temporally fit in the *R-Frame*; that is, each *R-Frame* must be of length $kT, k \in Z, k \neq 0$. Associated with each *R-Frame* is a *D-Frame*, which is the source of the *R-Frame's* domain blocks. A *D-Frame* must end at the same temporal location as its associated *R-Frame*, but may start before the beginning of that *R-Frame*. The length of a *D-Frame* is restricted to lM_3T frames, where $l \in Z, l \neq 0$. The *R-Frame* starting at time t is denoted by *R-Frame*(t), while the associated *D-Frame* is denoted *D-Frame*(t). Addresses within an *R-Frame* are relative to the last frame (which is also the last frame of the corresponding *D-Frame*), and increase along the negative temporal axis.

Using parameter values $k = 2, l = 4, M_1 = M_2 = 2, M_3 = 1$, and $T = 4$, the partitioning of an input signal into *R-Frames* and an example *R-Frame* with its associated *D-Frame* are illustrated in Fig. 1. Note that, for this selection of parameters, the *D-Frame* contains pixels which are outside of the corresponding *R-Frame*. It is therefore possible that a range

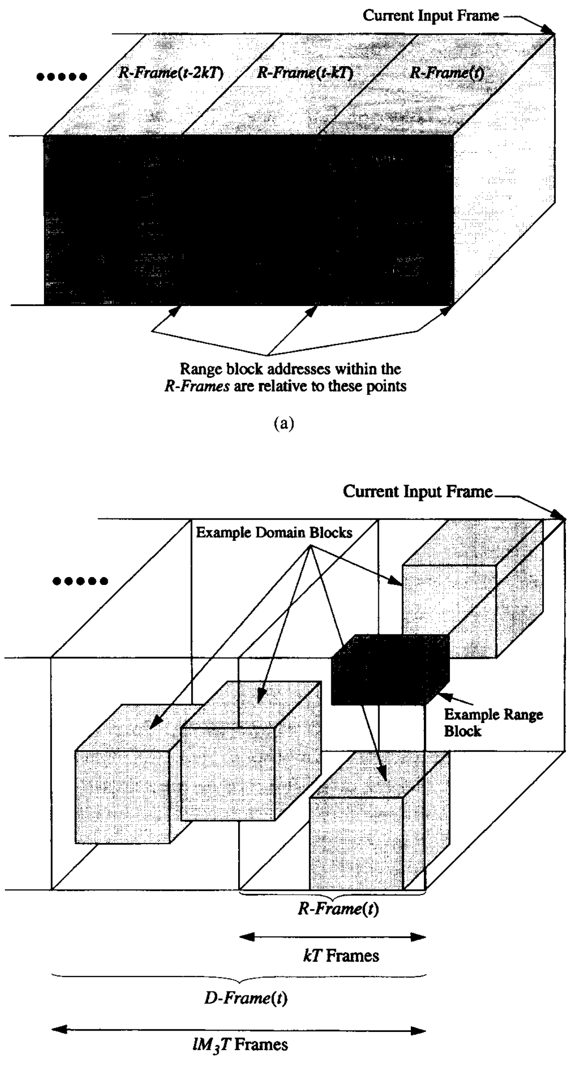

Fig. 1. (a) Partitioning of Input Sequence into *R-Frames*.(b) Example *R-Frame* and associated *D-Frame*.

block, r_i, be mapped from a domain block which is outside of its corresponding *R-Frame*. An advantage of this approach is that, for such range blocks, a single iteration can be used for decoding. Note that, for these parameters, the fractal code is no longer self-similar at the *R-Frame* level, although it is self-similar when viewed at the signal level. The disadvantage to this parameter selection is that the domain block search space is large, and thus encoding time is increased and the compression rate decreased when compared to parameters yielding a smaller *D-Frame*.

B. Range Block Mapping

As stated earlier, we adopt the form of range block mapping as given in (7). In this section, the various mapping parameters used for the proposed 3-D FBC are described.

Spatial Scaling Function For simplicity, pixel averaging over $M_1M_2M_3$ pixels is used for "spatial" contraction operator $S(\bullet)$. If the pixel value at location (x, y, z) within a block q is denoted by $q(x, y, z)$, then $S(\bullet)$ can be written as

$$S(q(x,y,z)) = \frac{1}{M_1M_2M_3}\sum_{a=0}^{M_1-1}\sum_{b=0}^{M_2-1}\sum_{c=0}^{M_3-1} q(x+a, y+b, z+c), \tag{11}$$

where it is understood that if $M_i < 1$, no summation in the i^{th} dimension occurs. In creating a spatially scaled block, (11) is applied to every M_i^{th} pixel in the i^{th} dimension within q.

Isometry Operation (Pixel Shuffling): For 3-D blocks, considerably more pixel shuffling operations are possible than for 2-D blocks. However, many such operations are not meaningful because, for most practical video signals, pixel values generally change smoothly through time. In order to keep the number of isometries to a reasonable value (and in keeping with the philosophy adopted earlier), isometries are restricted to operate on either an intra-frame basis, where the pixels within frames are shuffled, or on an inter-frame basis, where the order of the frames themselves are shuffled. We can therefore express the isometry operation, $I(i)$, in (7) as a combination of two isometries, $I(i) = I_{\text{inter}}(i) + I_{\text{intra}}(i)$, where $I_{\text{inter}}(i)$ and $I_{\text{intra}}(i)$ represent the inter-frame and intra-frame shuffling isometries, respectively.

The intra-frame isometries that we use, eight in total, are the same as those described by Jacquin in [5], while only two inter-frame isometries are used: an identity operation, in which the frame ordering remains unaltered, and a time-reverse operation, in which the order of the frames is reversed. Using these isometries, one bit is therefore required to identify $I_{\text{inter}}(i)$, and three bits are required to identify $I_{\text{intra}}(i)$.

Domain Block Search Method: The speed of the encoding algorithm is basically limited by the determination of the domain block matching function, $N(i)$. For 3-D coding, the number of possible blocks within the search space (i.e. the *D-Frame*) makes a full search of the space prohibitive. Thus, an efficient search strategy must be used.

As stated earlier, a number of search strategies have been proposed for use in the 2-D FBC algorithm. In this contribution, we restrict our attention to using the 3-D extension of the local search, in which domain blocks originating "near" to the range block being encoded are considered as possible candidates for $N(i)$ [12]. Specifically, if the address of the range block within the *D-Frame* is (n_1, n_2, n_3) then only domain blocks whose addresses are given by

$$(n_1 + k_1L_1, n_2 + k_2L_2, n_3 + k_3L_3), \quad -K_i \le k_i < K_i, \quad i = 1,2,3, \tag{12}$$

are searched.[2] The values (L_1, L_2, L_3) are search step-sizes, while the values (K_1, K_2, K_3) determine the size of the search region. Note that, since (L_1, L_2, L_3) and (K_1, K_2, K_3) are

[2]Clearly, (12) will yield some invalid search addresses for those range blocks near the boundaries of the *R-Frame*. In such cases, no alternate test is used during searching, thus limiting the domain pool for boundary blocks.

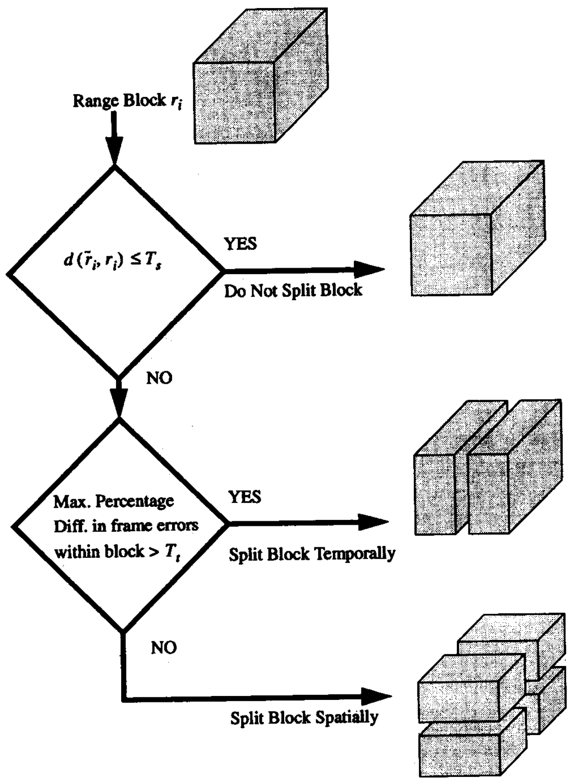

Fig. 2. Flowchart Description of Spatio-Temporal Range Block Splitting Method.

fixed for all *R-Frames*, this scheme has the advantage that only the values of (k_1, k_2, k_3) are required to address a domain block, with each k_i using $\lceil \log_2(2K_i) \rceil$ bits to encode ($\lceil a \rceil$ indicates the smallest integer no larger than a).

Spatio-Temporal Range Block Splitting: As mentioned earlier, a quad-tree approach is often used to split range blocks when encoding errors are too high. In addition to using 3-D quad-tree partitioning, known as oct-tree partitioning, we introduce a novel method known herein as spatio-temporal partitioning. In spatio-temporal partitioning a range block is split either spatially, by 4, or temporally, by 2, depending upon the distribution of errors within the original range block and upon the overall encoding error. A flowchart description of the spatio-temporal partitioning algorithm is given in Fig. 2, and is described below.

If the overall block encoding error, $d(\tilde{r}_i, r_i)$, is beyond the threshold, T_s, *and* the errors are distributed evenly throughout the frames of the range block, then the block is split spatially into four blocks whose depths remain unchanged. On the other hand, if the encoding error for the block exceeds the threshold T_s *and* the errors are distributed unevenly within the blocks' frames, the block is split temporally into two range blocks which have the same spatial size but which are half the depth of the original block.

The error between the maximum and minimum frame encoding errors, normalized by the maximum frame encoding error, is used as an indication of the error distribution within a range block. The frame encoding errors are determined by computing the distance between each frame from the encoded block, $\tilde{r}_i$, and the original frames from the block r_i. If the normalized difference between the maximum and minimum of these distances is beyond a threshold value T_t, the error distribution is deemed to be "uneven", otherwise, the errors are assumed to be distributed evenly throughout the block.

When using spatio-temporal partitioning, we denote the maximum number of spatial splits a block may undergo by N_s, and the maximum number of temporal splits a block may undergo by N_t. Therefore, assuming that a predetermined ordering of range blocks is used when storing the transformations for an *R-Frame*, each range block requires $\lceil \log_2(N_s + 1) \rceil + \lceil \log_2(N_t + 1) \rceil$ bits to encode its partition level. (This number can be reduced by using a more sophisticated range block ordering scheme. We do consider this possibility here.) For 3-D oct-tree partitioning, each block requires $\lceil \log_2(N_O + 1) \rceil$ bits to encode the partition level.

C. Algorithm Pseudo-Code

Using the above mapping parameters and range block partitioning method, a pseudo-code description of the proposed 3-D fractal block coding algorithm is given (see top of next page).

D. Considerations in Decoding

Decoding a fractal block code consists of iteratively applying the transformation associated with each range block until there is sufficiently little difference between the output from successive iterations [5]. For the input data partitioning scheme presented here, each *R-Frame* is decoded by iteratively applying the corresponding transformations a fixed number of times. Note that in order to decode the *R-Frame*, all frames corresponding to the appropriate *D-Frame* are required. If the *D-Frame* parameters are such that the *D-Frame* contains data outside of the corresponding *R-Frame*, as in the example presented earlier, this data will also be required during decoding (but will not be iterated since it is outside the *R-Frame*). As stated earlier, range blocks whose corresponding domain block is completely contained in such an area, can be decoded in a single iteration.

Similar to all FBC decoding schemes, the number of decoding iterations is dependent upon how close the initial values of the *R-Frame* are to the fixed point of its transformation. In 2-D (image) coding, it is typical to start with a zero valued (all black) image as the initial point for decoding. However, in 3-D coding, the fact that there is often little change over several frames (particularly in video conferencing applications) can be used to advantage. In the intra-frame coder proposed in [3], the previously decoded frame is used as the starting point for the next frame to be decoded. We adopt a similar approach; specifically, the first decoding iteration for *R-Frame*(t) is the same as the final iteration for *R-Frame*(t-kT). Results from varying the number of decoding iterations using this type of *R-Frame* preloading, are given in Section IV.

Pseudo Code Description of the 3-D Fractal Block Coding Algorithm

```
/* R-Frame processing */
while R-Frames to process
    initialize current R-Frame, R-Frame($t$), and associated D-Frame, D-Frame($t$);
    initialize range pool $\{r_i\}$ for R-Frame($t$) such that blocks are of size $B \times B \times T$;
    /* process each range pool block */
    for each range block $r_i$ in $\{r_i\}$ do
        best_transform = $\{\alpha_i = o_i = NULL, I(i) = NULL, N(i) = NULL\}$;
        best_error = infinity;
        /* perform local search */
        for all $d_j$ from D-Frame ($t$) satisfying equation (12) do
            for each possible intra-frame isometry, $I_{\text{intra}}$, do
            for each possible inter-frame isometry, $I_{\text{inter}}$, do
                compute parameters $\alpha$ and $o$ for current $I_{\text{intra}}$, $I_{\text{inter}}$ and $d_j$;
                compute approximation block, $\tilde{r}_i$, from mapping parameters;
                if $d(\tilde{r}_i, r_i) <$ besterror then /* found a better block */
                  best_transform = $\{\alpha = \alpha_i, o_i = o, I(i) = I_{\text{inter}} + I_{\text{intra}}, N(i) = j\}$;
                  best_error = $d(\tilde{r}_i, r_i)$;
                endif
            endfor /* intra-frame isometries */
            endfor /* frame-based isometries */
        endfor /* local search */
        /* decide if block is to be split */
        if best_error $< T_s$ then
            accept best_transform as transform for range block $r_i$;
        else /* shown below is the spatio-temporal splitting algorithm */
            /* compute approximation error for each frame from $r_i$, and
                compute maximum normalized difference */
            max_framediff = MaxFrameerror($\tilde{r}_i$)−MinFrameerror($\tilde{r}_i$);
            normalized_diff = max_framediff / MaxFrameerror($r_i$) * 100.;
            if normalized_diff $> T_t$ AND NumTemporalSplits($r_i$) $\leq N_T$ then
                split $r_i$ into 2 range blocks, each one half the depth of $r_i$;
                increment NumTemporalSplits for the newly split blocks;
                add newly split blocks to $\{r_i\}$;
            else if NumSpatialSplits($r_i$) $\leq N_s$ then
                split $r_i$ into 4 range blocks, each one quarter the spatial size of $r_i$;
                increment NumSpatialSplits for the newly split blocks;
                add newly split blocks to $\{r_i\}$;
            else /* maximum number of splits for this block */
                accept best_transform as transform for range block $r_i$;
            endif
        endif /* block to be split */
    endfor /* range block processing for current R-Frame */
endwhile /* R-Frame processing */
```

IV. Results and Discussion

In this section, results are presented from the encoding of two "standard" video sequences, representative of video conferencing data (i.e. the sequences do not contain significant camera movement such as panning and zooming). We begin by presenting how the compression rate of the proposed algorithm is computed, followed by example parameter values and associated encoding results.

A. Computation of the Compression Rate

One of the advantages of the FBC decoding algorithm is that it is possible (due to the spatial contraction operation $S(\bullet)$) to create simulated detail at a higher resolution than of the input data. It has been claimed that creating such detail increases the "effective" compression rate of the FBC algorithm [2] [16]. We consider any additional detail created at resolutions higher than that of the original image to be "interpolated" data, which, for the purposes of this paper, are not counted in any compression rate calculations. (Although it is possible to create this simulated detail with the methods described herein, we only present decoding results which are at the resolution of the original input data.)

The compression rate of the proposed 3-D FBC algorithm depends upon several factors: the number of range blocks per

TABLE I
PARAMETER VALUES USED FOR 3-D FBC ENCODING

Parameter	Value			Comments
	Test 1	Test 2	Test 3	
B	8	8	8	Original spatial rangle block size (pixels)
T	8	8	8	Original temporal range block size (frames)
k	1	1	1	*R-frame* depth factor; yields *R-Frames* 8 frames deep
l	1	1	1	*D-Frame* depth factor; yields *D-Frames* 8 frames deep
N_o	N/A	1	N/A	Number of oct-tree splits
N_s	1	N/A	1	Maximum number of spatial splits for a range block using spatio-temporal splitting
N_t	3	N/A	3	Maximum number of temporal splits for a range block using spatio-temporal splitting
T_s	0.5	0.5	0.5	Block splitting threshold value - used as both oct-tree threshold and spatial threshold in spatio-temporal splitting
T_t	17%	N/A	17%	Temporal Split threshold value
$L_1 = L_2$	4	4	1	Spatial search step size (pixels)
L_3	1	1	1	Temporal search step size (frames)
$K_1 = K_2$	2	2	1	Spatial search size
K_3	1	1	1	Temporal search size
$M_1 = M_2$	2	2	2	Domain block scaling value - spatial
M_3	1	1	1	Domain block scaling value - temporal
α	Max: 1.5, Min: -1.5	Max: 1.5, Min: -1.5	Max: 1.5, Min: -1.5	Limits of scaling value used in range block mapping (uniformly quantized)
α_B	5	5	5	Number of bits used to represent scaling value
o	Max: 255, Min: -253	Max: 255, Min: -253	Max: 255, Min: -253	Range of offset value used in range block mapping (uniformly quantized)
o_B	6	6	6	Number of bits used to represent offset value

R-Frame (which is, in turn, dependent upon the multi-level partitioning parameters, original range block size and input data); the *R-Frame* and *D-Frame* sizes; and the number of bits required for each range block transformation.

Using spatio-temporal partitioning and the parameter values from the previous section, the number of bits, B_i, required to encode the transformation for each range block is given by

$$B_i = \alpha_B + o_B + I_B + \lceil \log_2(N_s + 1) \rceil + \lceil \log_2(N_t + 1) \rceil + \sum_{i=1}^{3} \lceil \log_2(2K_l) \rceil \tag{13}$$

where α_B and 0_b represent the number of bits required to store the scaling and offset values, respectively, I_B and represents the number of bits required to identify the inter-frame and intra-frame isometries. For the isometries used herein, and described earlier, $I_B = 4$. Using the oct-tree partitioning method, B_i is given by

$$B_i = \alpha_B + o_B + I_B + \lceil \log_2(N_0 + 1) \rceil + \sum_{l=1}^{3} \lceil \log_2(2K_l) \rceil. \tag{14}$$

If the number of range blocks required to encode *R-Frame*(t) is denoted by $N_R(t)$, the number of bits needed to encode *R-Frame*(t) can be expressed by

$$R\text{-}Frame(t)_B = N_R(t) B_i, \tag{15}$$

while the number of bits-per-pixel (bpp) for *R-Frame(t)* is given by

$$R\text{-}Frame(t)_{\text{bpp}} = \frac{N_R(t) B_i}{kTN_1N_2}, \tag{16}$$

where N_1 and N_2 denote the number of width and height of the input frames, respectively. Note that $N_R(t)$ is constant valued for each group of kT frames (corresponding to the length of each *R-Frame*). The compression rate within each *R-Frame* can found by dividing the number of bits originally used to represent each pixel by *R-Frame*$(t)_{\text{bpp}}$.

B. Coding Results

The first 144 frames, of size 360 × 288 pixels, from the standard "Salesman" and "Miss America" 8-bit greyscale video sequences have been encoded using the proposed 3-D FBC algorithm using three sets of encoding parameters, referred to as "Test 1", "Test 2", and "Test 3". The parameter values corresponding to these test modes are provided in Table 1. In "Test 1" and "Test 3", spatio-temporal range block splitting has been used, with "Test 3" employing a more restrictive domain block search-space than "Test 1". In "Test 2", similar parameters to "Test 1" have been used but with range block splitting performed using the oct-tree method. We make no claim as to any optimal nature of these coefficients; our experience has shown that the parameters leading to the best compression and highest quality reconstruction are dependent upon the nature of the video sequence being encoded.

Decoded frames 107–109 from the "Salesman" "Test 1" sequence, and decoded frames 75–77 from the "Miss America" "Test 1" sequence are shown in Fig. 3 and Fig. 4, respectively. Detailed sections of Frame 107 from the "Salesman" "Test 1" sequence, and of Frame 75 from the "Miss America" "Test 1" sequence are shown in Fig. 5. Six iterations of the decoding algorithm have been used in both cases, with the initial values

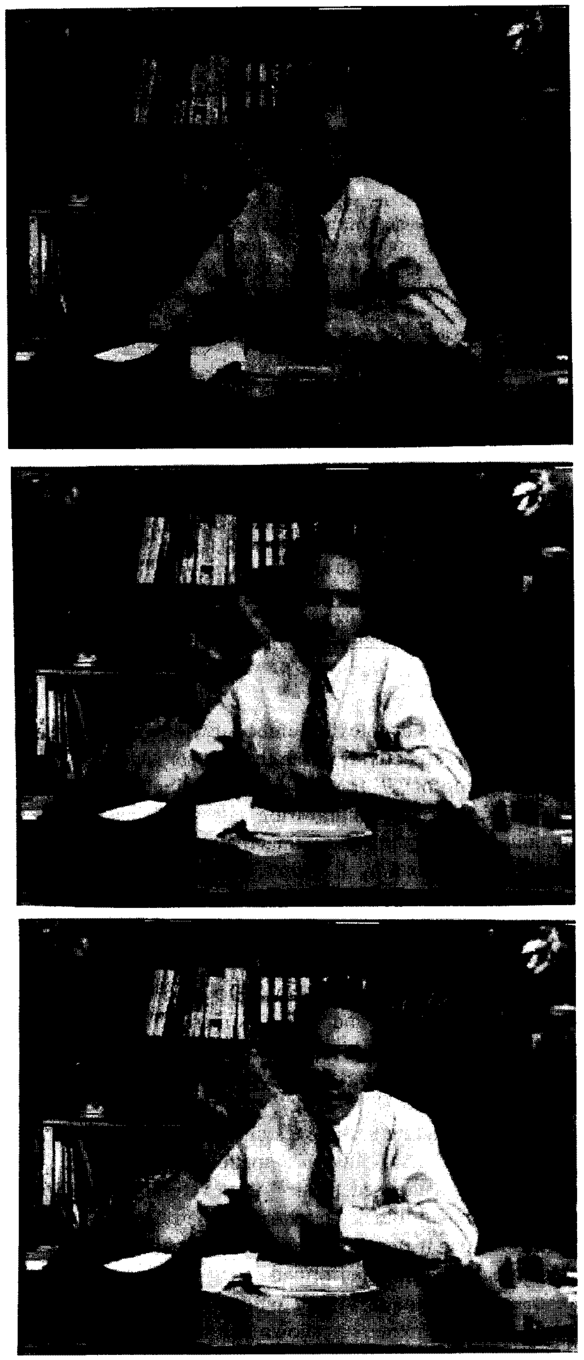
Fig. 3. Decoded Frames 107–109 from "Salesman", "Test 1" Parameters, 6 Decoding Iterations.

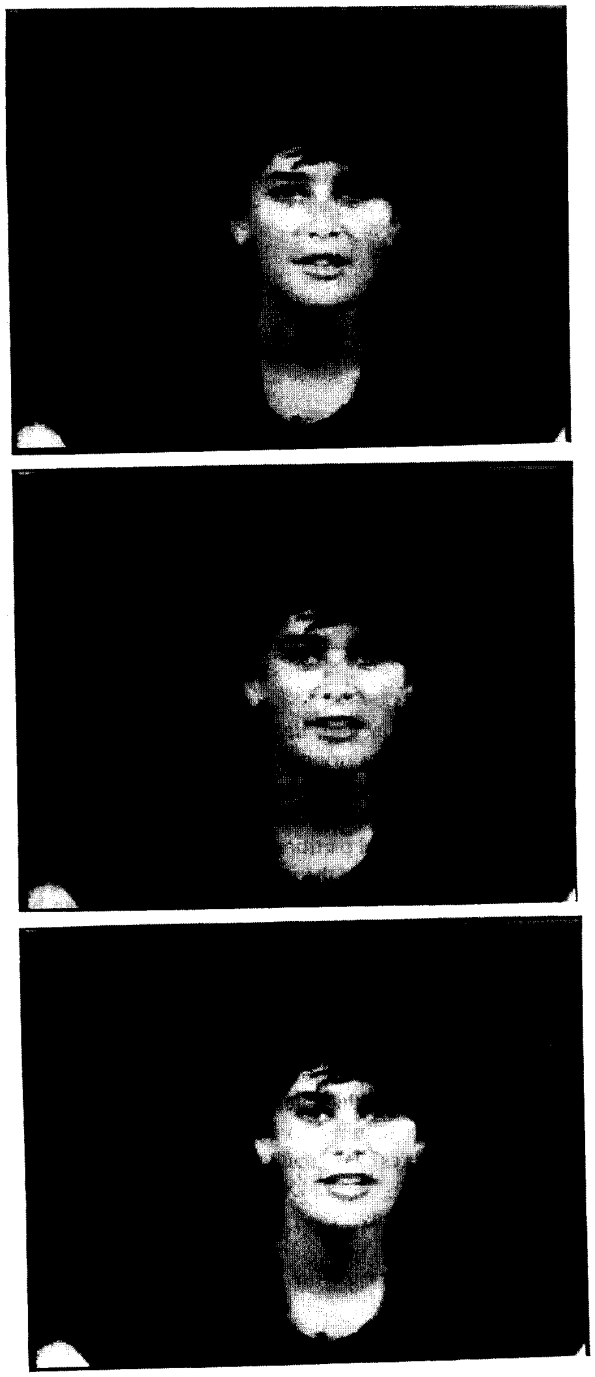
Fig. 4. Decoded Frames 75–77 from "Miss America", "Test 1" Parameters, 6 Decoding Iterations.

of the *R-Frame* preloaded with the results from the previous *R-Frame's* decoding sequence, as described earlier. The initial pixel values for R-Frame(0) have been set to zero (i.e. "all black").

The compression rates and the peak-to-peak signal to noise ratio (PSNR) are shown in Fig. 6. The PSNR between two signals, a, and $\tilde{a}$ has been computed using

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{255^2}{\mathrm{MSE}(a,\tilde{a})}\right) \tag{17}$$

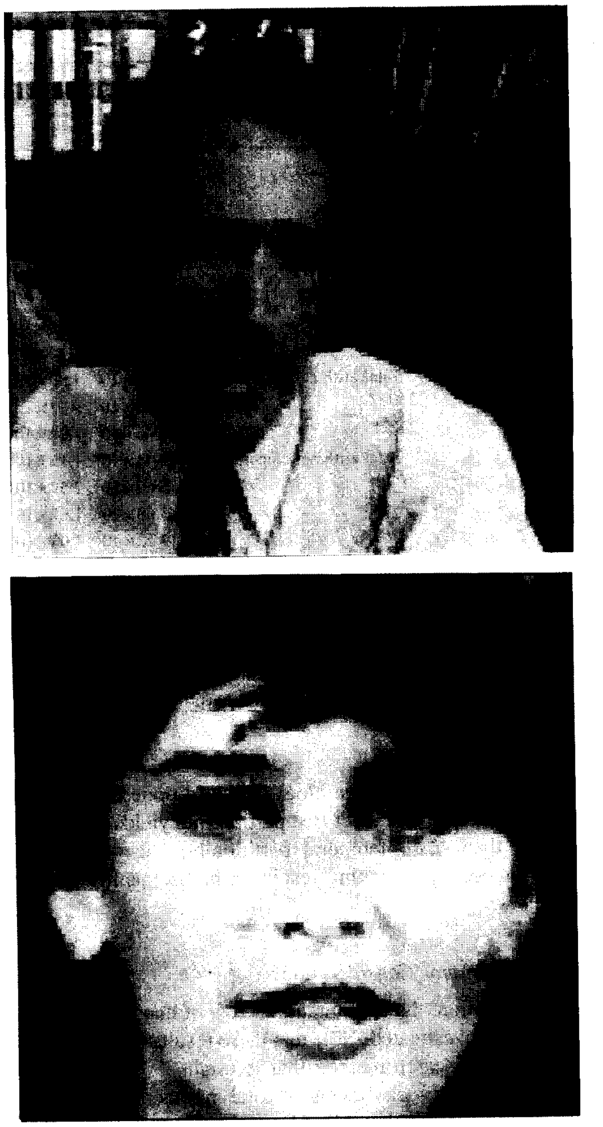
Fig. 5. Detail from Decoded Frame 107 from "Salesman" in Fig. 3 and from Decoded Frame 75 from "Miss America" in Fig. 4.

where MSE$(a, \tilde{a})$ represents the mean squared between the signals a and $\tilde{a}$. The effects of varying the number of decoding iterations upon the PSNR of the "Salesman" "Test 1" sequence, while still using an *R-Frame* preloading scheme, is shown in Fig. 7.

From the results shown in Figs. 3–5, it is clear that, although there are noticeable artifacts present in the "Test 1" decoded frames, the subjective quality of reconstruction is sufficient for many video-coding applications, including video-conferencing. When the corresponding sequences are played back at video rates, these artifacts manifest themselves as irregular or jerky motion and are most noticeable in those frames which contain significant object motion. The

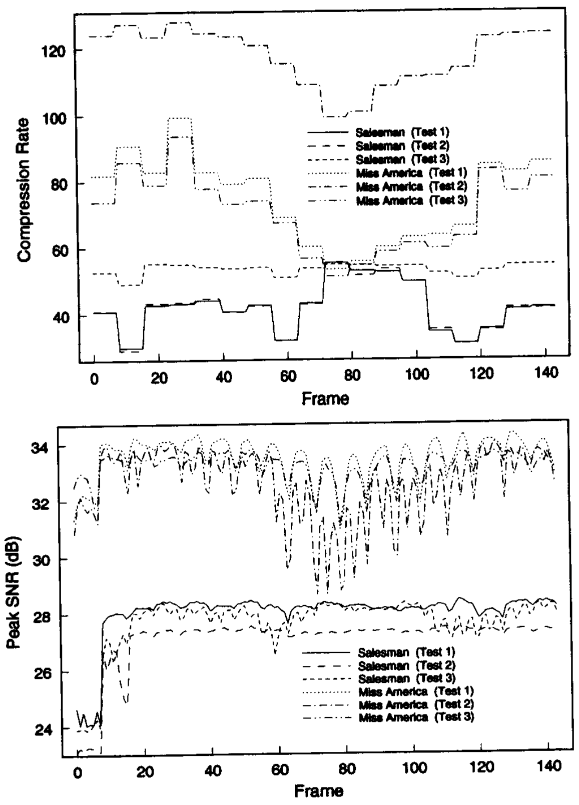

Fig. 6. Compression Rate and PSNR for "Salesman" and "Miss America" Sequences.

corresponding results for the "Test 2" sequences also contain these motion artifacts. However, in the "Test 2" sequences, these artifacts are more noticeable and consequently more perceptually annoying. The results for the "Test 3" sequences are similar to the "Test 1" sequences, but contain somewhat more noticeable blocking. In addition, the "Test 3" sequence of "Salesman" occasionally contains areas which have "saturated" that is, which are all white. We conjecture that the reduced search space used for the "Test 3" sequences results in some domain block mappings which are not eventually contractive and hence causes this saturation artifact.

The compression rates, shown in Fig. 6, clearly demonstrate the effects of the multi-level partitioning, as indicated by the different compression rates obtained for various *R-Frame*. Therefore, because the bit-rate is related to the compression rate, a buffered decoder would be required for fixed-bandwidth applications of this algorithm.

The effect of varying the number of decoding iterations upon the PSNR of a reconstructed sequence, as shown in Fig 7, illustrates two points. The first, is that increasing the number of decoding iterations past five or six iterations yields little increase in the overall PSNR of the reconstructed sequence. It is for this reason that we have used six decoding iterations for reconstruction. The second point is in relation to the

TABLE II
DETAILS OF ENCODING FOR $R - Frame(0)$ FOR "TESTS 1-3" OF "MISS AMERICA"

Test	RRange block size - number of pixels (N)	Number of range blocks tested	Number of range blocks accepted	Number of domain blocks tested	Computational Cost (Adds, Mults)
1	8x8x8 - 512	1620	1487	24012	$(2.16\times10^9, 9.83\times10^8)$
	8x8x4 - 256	152	2	3555	$(1.60\times10^8, 7.28\times10^7)$
	8x8x2 - 128	164	0	4464	$(1.00\times10^8, 7.28\times10^7)$
	8x8x1 - 64	66	0	1904	$(2.14\times10^7, 9.75\times10^6)$
	4x4x8 - 128	228	196	2475	$(5.58\times10^7, 2.53\times10^7)$
	4x4x4 - 64	336	72	7923	$(8.92\times10^7, 4.06\times10^7)$
	4x4x2 - 32	1052	595	29083	$(1.64\times10^8, 7.45\times10^7)$
	4x4x1 - 16	1178	1178	34681	$(9.76\times10^7, 4.44\times10^7)$
				Total arithmetic operations =	$(2.90\times10^9, 1.30\times10^9)$
2	8x8x8 - 512	1620	1487	24012	$(2.16\times10^9, 9.83\times10^8)$
	4x4x4 - 64	1064	1064	21828	$(2.46\times10^8, 1.12\times10^8)$
				Total arithmetic operations =	$(2.41\times10^9, 1.10\times10^9)$
3	8x8x8 - 512	1620	1457	6319	$(5.69\times10^8, 2.59\times10^8)$
	8x8x4 - 256	136	0	816	$(3.68\times10^7, 1.67\times10^7)$
	8x8x2 - 128	136	0	816	$(2.22\times10^7, 1.00\times10^7)$
	8x8x1 - 64	50	0	984	$(4.46\times10^6, 2.03\times10^6)$
	4x4x8 - 128	380	254	1329	$(2.99\times10^7, 1.36\times10^7)$
	4x4x4 - 64	524	149	3016	$(3.40\times10^7, 1.54\times10^7)$
	4x4x2 - 32	1194	763	8372	$(4.72\times10^7, 2.14\times10^7)$
	4x4x1 - 16	1062	1062	8084	$(2.28\times10^7, 1.03\times10^7)$
				Total arithmetic operations =	$(7.66\times10^8, 3.48\times10^8)$

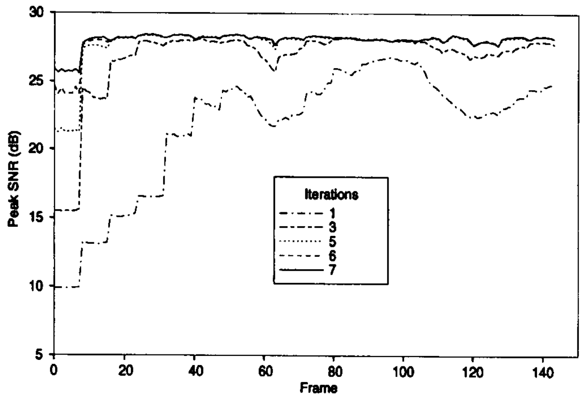

Fig. 7. The effect of varying the number of decoding iterations upon the PSNR for the "Test 1" encoded sequence of "Salesman".

preloading of the *R-Frames*. Specifically, the average PSNR value for *R-Frame*(0), corresponding to the first $kT = 8$ frames, is consistently lower than all other *R-Frames* for the number of decoding iterations tested. In fact, when less than five decoding iterations are used, the PSNR values for the first few *R-Frames* are typically lower than those for the subsequent *R-Frames*. This indicates that preloading of the *R-Frame* is an important factor in reducing the number of decoding iterations.

In general, the results presented here are consistent with other proposed FBC algorithms, and indicate that the encoding quality is dependent upon the size of the domain pool and the manner in which range blocks are partitioned. The results also indicate that, while oct-tree partitioning yields better compression than spatio-temporal partitioning, the flexibility of spatio-temporal partitioning results in higher quality encoded sequences.

C. Computational Considerations

In this section, the computational cost of encoding a typical *R-Frame* is investigated. Note that these results reflect the local domain block search method that is used.

For a range block having a total number of pixels N, there are approximately $7N$additions and $4N$ multiplications required to evaluate α_i, o_i, and $d(r_i, \bar{r})$ for each tested spatially contracted domain block. In order to obtain a spatially contracted domain block, $S(d_j)$, (11) is applied to $d_j N$ times, which introduces an additional $M_1 M_2 M_3 N$ additions and N multiplications (we assume that the division operation in (11) is computationally equivalent to a multiplication). For the values of M_i used herein, this yields a total of $11N$ additions and $5N$ multiplication operations. Finally, since we use a total of sixteen isometry operations, *each* domain block tested as a possible candidate for $N(i)$ requires a total of $C_A = 176N$ additions and $C_M = 80N$ multiplications.

In Table II, encoding results for *R-Frame*(0) using all three test modes are summarized for the "Miss America" sequence. Included in Table II are the number of range blocks tested and accepted at each partition level, the total number of domain blocks tested as candidates for $N(i)$ at each partition level, and the total number of arithmetic operations required to encode the *R-Frame*.

The results from Table II reflect several factors about the proposed 3-D FBC algorithm and the various test modes used. First, the oct-tree range block splitting algorithm ("Test 2") requires slightly fewer arithmetic operations and yields higher compression that corresponding spatio-temporal algorithm ("Test 1"). However, the cost of this higher compression is lower reconstruction quality, both in terms of PSNR and subjective quality. Next, and as expected, the number of arithmetic operations from the "Test 3" sequence, which yield a reduced domain block search space, are fewer than for either the "Test 1" or "Test 2" sequences. It is interesting to note that the reconstruction quality of the "Test 3" sequence is only slightly lower than for the "Test 1" sequence, but higher than for the "Test 2" sequence. Finally, it is clear from Table 2 that the FBC encoding algorithm is computationally intensive. (For example, the average number of additions and multiplications per pixel for the "Test 3" sequence are 924 and 420, respectively.) Clearly, in order to use FBC for real-time systems, a more efficient domain block search strategy is required.

The computational cost of decompression is primarily dependent upon the number of decoding iterations used. For the parameters selected herein, it can be shown that each pixel requires 5 additions and 2 multiplications per iteration. Assuming six decoding iterations are sufficient, it is clear that decoding can be performed considerably faster than encoding and that real-time decompression is possible.

D. Recommendations for Improving the Algorithm Performance

For the "Test 1" "Salesman" and "Miss America" sequences, the average compression rates are 41.80 and 74.39, respectively. Assuming a frame rate of thirty frames per second, that an average compression rate of 40 is possible for typical video sequences (using a buffered decoder), and that the compression can be performed in real-time, the number of 8-bit (grey-level) pixels which can be transmitted at 64 kbits/second is 10.67×10^3, which is roughly equivalent to a frame size of 120×80 pixels. *For very low-bit rate applications*, it is therefore advantageous to further increase the compression achieved by the proposed 3-D FBC algorithm.

Two principal ways in which this can be achieved are by entropy encoding the transformations for each *R-Frame* and by using "carry forward" or "constant" blocks [3]. By entropy encoding the FBC coefficients for each *R-Frame*, any structure (that is, correlation) inherent within the transformations can be removed, resulting in a lower bit-rate or higher compression rate. (We have observed that the parameter values within *R-Frame* transformations do exhibit some structure, particularly with respect to the inter-frame isometry operations).

Constant blocks can be further used to represent those range blocks within an *R-Frame* which do not change significantly through time. In other words, if there are a set of range blocks representing a constant background over several frames, only the transformation for one of the corresponding range blocks would be required with the other range blocks using the same code.

V. Conclusion

In this paper, the basic fractal block coding algorithm has been reviewed. A novel three-dimensional fractal block coding algorithm has been proposed in which the input video stream is partitioned into *R-Frames*, from which range blocks are obtained, and *D-frames*, from which the associated domain blocks are selected. This method of partitioning permits domain block search methods, developed for image encoding, to be used in three-dimensional fractal coding. The proposed method utilizes three-dimensional range and domain blocks, which allows for higher compression than frame by frame video compression. In addition, a novel spatio-temporal range block splitting mechanism is described in which the decision to split a range block is based upon the overall encoding error as well as the distribution of errors within the approximated block.

After presenting the details of the proposed algorithm, encoding and decoding results, obtained from two standard video test sequences, are given. The results indicate that average compression rates of 40 to 77 can be achieved with subjective quality suitable for video-conferencing applications. However, the high computational cost of the encoding algorithm and the approximately fixed cost of decoding suggests that the proposed algorithm may be better suited to asymmetric applications, such as multi-media. In order to make 3-D fractal block coding suitable for very low bit rate systems, the use of entropy encoding and constant (or carry-forward) blocks would be advantageous.

Acknowledgment

The authors acknowledge funding for this work by MICRONET, the (Canadian) Federal Network of Centres of Excellence on Microelectronics, Devices and Systems, the Natural Sciences and Engineering Research Council of Canada, and the University of Calgary through their Silver Anniversary Scholarship program. The authors also thank the reviewers for their helpful comments and suggestions.

References

[1] M. F. Barnsley, *Fractals Everywhere*, Academic Press, 1988.
[2] M. F. Barnsley and L. P. Hurd, *Fractal Image Compression*, AK Peters: Wellesley, Massachusetts, 1993.
[3] L. P. Hurd, M. A. Gustavus and M. F. Barnsley, "Fractal video compression," *37th Annual IEEE International Computer Conference (COMPCON)*, pp. 41–42, 1992
[4] M. F. Barnsley and A. Sloan, United States Patent #5065447, "Method and apparatus for image compression," 1991.
[5] A. Jacquin, "Image coding based on a fractal theory of iterated contractive image transformations," *IEEE Trans. Image Processing*, vol. 1, no. 1, pp. 18–30, Jan. 1992.
[6] ——, "Fractal Image Coding: A Review," *Proceedings of the IEEE*, vol. 81, no. 10, pp. 1451–1465, Oct. 1993.
[7] M. S. Lazar and L. T. Bruton, "Efficient fractal block coding using multiresolution decomposition based search constraints," University of Calgary, Department of Electrical and Computer Engineering, *Technical Report*, October 1993.
[8] G. Øien, S. Lepsoy and T. Ramstad, "An Inner product space approach to image coding by contractive transformations," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Toronto, pp. 2773–2776, May 1991.
[9] ——, "Reducing the complexity of a fractal-based image coder," *Signal Processing VI: Theories and Applications*, J. Vandewalle, R.

Boite, M. Moonen and A. Oosterlinck (eds.), Elsevier Science Publishers B.V., pp. 1353–1356, 1992.

[10] S. Lepsoy, G. Øien and T. Ramstad, "Attractor image compression with a fast non-iterative decoding algorithm," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Minneapolis, vol. 5, pp. 337–340, Apr. 1993.

[11] M. Gharavi-Alkhansari and T. S. Huang, "A Fractal-based image block coding algorithm,"*IEEE International Conference on Acoustics, Speech, and Signal Processing*, Minneapolis, vol. 5, pp. 345–348, Apr. 1993.

[12] G. Vines and M. H. Hayes III, "Adaptive IFS image coding with proximity maps," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Minneapolis, vol. 5, pp. 349–352, Apr. 1993.

[13] L. Thomas and F. Deravi, "Pruning of the transform space in block-based fractal image compression," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Minneapolis, vol. 5, pp. 341–344, Apr. 1993.

[14] D. M. Munro, "A hybrid fractal transform," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Minneapolis, vol. 5, pp. 169–172, Apr. 1993.

[15] Y. Fisher, "A Discussion of fractal image compression," Appendix A in *Chaos and Fractals: New Frontier of Science*, H. Peitgen, H. Jurgens and D. Saupe (eds.), New York: Springer-Verlag, pp. 903–919, 1992.

[16] ——, "Fractal image compression," SIGGRAPH 1992 Course Notes.

[17] Y. Fisher, E. W. Jacobs and R. D. Boss, "Fractal Image Compression Using Iterated Transforms," in *Image and Text Compression*, J. A. Storer (ed.), Dordrecht, Netherlands: Kluwer, pp. 35–61, Dec. 1992.

[18] E. W. Jacobs, Y. Fisher and R. D. Boss, "Image Compression: A study of the iterated transform method," *Signal Processing*, vol. 29, pp. 251–263, Dec. 1992.

[19] D. Le Gall, "MPEG: a video compression standard for multimedia applications," *Communications of the ACM*, vol. 34, pp. 46–58, 1991.

Michael Lazar received a B.Sc. in Computer Engineering from the University of Alberta in 1984 and an M.Sc. in Electrical Engineering from the University of Alberta in 1988.

Currently, he is working toward his Ph.D. in Electrical Engineering at the University of Calgary, Canada. He has also worked for Bell Northern Research from 1985–1986 and 1988–1990. His research interests include image and video processing, multiresolution analysis and the applications of fractals within engineering.

Leonard T. Bruton (M'71–SM'80–F'81) is a Professor of Electrical and Computer Engineering at The University of Calgary, Calgary, Alberta, Canada.

His research interests are in the areas of analog and digital signal processing. He is particularly interested in the design and implementation of microelectronic digital filters and the applications of multidimensional circuit and systems theory to digital image processing.