THE UNIVERSITY OF
CALGARY

FACULTY OF ENGINEERING
Department of Electrical and Computer Engineering

Efficient Fractal Block Coding Using Multiresolution Decomposition Based Search Constraints

by

M. S. Lazar and L. T. Bruton

MD-DSP Group Technical Memorandum
Report #TM-001.93
April 20, 1993

Efficient Fractal Block Coding Using Multiresolution Decomposition Based Search Constraints

M. S. Lazar and L. T. Bruton
University of Calgary, Calgary Alberta, Canada

Abstract

Fractal block-based image coders have received considerable attention recently [3][6][7]. These coders provide high compression rates, fast decoding algorithms, good image reconstruction, and can supply simulated detail not found in the original image. An undesirable property of many fractal block-based coding algorithms, however, is the large associated encoding time which is due predominately to an extensive block matching full search [7] that occurs for each image block. In this contribution, we present two methods which reduce the encoding time of the fractal block-based coders by constraining the search performed for each block. The first method we propose consists of encoding lower resolution images which are derived from the original image. The results obtained from a low resolution image are used to constrain the search of each corresponding block at the next highest resolution. This process is repeated until the constraints have been propagated to the encoding of the image at the original resolution. The second method is based upon searching a small randomly selected subset of the total search space. We present results obtained using a fractal block-based coding algorithm employing the proposed constraint search mechanisms. We further show that the overall encoding time is reduced when compared to the full search algorithm [7] and that comparable compression rates and coding quality are maintained.

I. Introduction

Unlike the iterated function system (IFS) scheme [1], which relies upon global-self similar properties of an image, fractal block image coders (FBCs) rely on self-similarity between image regions in what can be described as piece-wise self-similarity. Much of the seminal work on fractal block coders has been published by A. Jacquin [2]-[5], with several variations and improvements subsequently proposed [6][7]. The encoding processes in all FBCs, however, share similar characteristics, which follows.

In a similar fashion to all block-coding algorithms, FBCs first require that the image to be encoded is partitioned into a set of blocks which form a non-overlapping covering of the image (the partition may, of course, change as encoding progresses). We denote blocks from this set, r_i, as *range* blocks, and the set $\{r_i\}$ as the *range pool*. The FBC algorithm encodes each range block by mapping it to another block, d_j, selected from a *domain pool*, $\{d_j\}$. This mapping consists of contrast scaling, an offset, pixel shuffling, and spatial contraction operations. The domain pool consists of possibly non-overlapping blocks obtained from the original image and can be much larger than the range pool. In general, the larger $\{d_j\}$ the better the resulting encoding and the longer the compression process. The fractal image code for the entire image consists of identifying, for each r_i, the corresponding d_j and associated mapping parameters.

Our experience has shown that the number of blocks in the domain pool affects the speed of the encoding process. Reducing the number of blocks in $\{d_j\}$ decreases decoding time, but limits the "richness" of the domain pool and results in poorer encoding quality. Conversely, increasing the number of blocks in $\{d_j\}$ increases coding time and results in better quality reconstruction. In this contribution, we present two methods which can be used to constrain the number of searches performed when mapping each r_i.

In the first method, here referred to as the Multiresolution Constraint Propagation (MCP) algorithm, the original image is decomposed into a set of low resolution images via a multiresolution decomposition, and then each image, starting with the lowest resolution image, is coded successively. The mapping for each r_i obtained at a particular resolution is used to limit the domain search area used for the corresponding r_i at the next highest resolution. Thus, a full search algorithm, required to start the constrained search process, is required only at the lowest resolution image and faster constrained searches can be used for the r_i at higher resolutions.

The second method, here referred to as the Random Constraint Search (RCS) algorithm, constrains the domain pool search to domain blocks which originate from a randomly selected area of the image. While, at least intuitively, the RCS method would seem to provide a poor selection of domain blocks from upon which to base the coding parameters, we have found that, when compared to either the conventional full search [7] approach or the above MCP method, reasonable coding quality and compression rates are realized with substantial savings in coding time.

This paper is organized as follows: in Section II we review the theoretical background of fractal block-based image coders and provide a multiresolution view of the full search coding algorithm; the MCP algorithm is presented in Section III; in Section IV the RCP algorithm is presented; in Section V we compare the various methods with regards to coding accuracy, compression bit rates, and coding time, with directions for future work also discussed; and, in Section VI, conclusions are presented.

II. Review of Fractal Block-Based Image Coding

The mathematical basis for fractal block coders (FBCs) rests upon the theory of iterated contractive transformations in metric spaces [1]. We briefly review the theory of fractal block coding which may be found, in more detail, in [1][2][7].

A. Properties of Metric Spaces

Consider the complete metric space, $(\Im, d)$, where $\Im$ is the space of discrete domain (i.e. sampled) images of size $N_1 \times N_2$, $(N_1, N_2 \in Z,\ Z = \{1, 2, \ldots\})$, such that $\Im \subset (Z \times Z \times R)$, and d is the (normalized) Euclidean distance between the *sample* values of two members of $\Im$. For $a, b \in \Im$, d is defined by

$$d(a, b) = \left(\frac{1}{N_1 N_2}\left(\sum_{i=1}^{N_1}\sum_{j=1}^{N_2}(a_{i,j} - b_{i,j})^2\right)\right)^{1/2} \in R \quad , \tag{1}$$

where $a_{i,j}, b_{i,j} \in R$, $(1 \le i \le N_1,\ 1 \le j \le N_2)$, represent the image sample values at pixel location (i, j).

For the space $\Im$, the mapping, $\Phi: \Im \to \Im$ is said to be contractive [1] if and only if (iff)

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

If $\Im$ is a complete metric space, then Banach's fixed point theorem [1] ensures that each contractive mapping in $\Im$ has a unique fixed point, p_f, such that

$$\Phi(p_f) = p_f \ . \tag{3}$$

Moreover, if Φ is contractive then Φ may be iteratively applied to *any* point, $q \in \Im$, yielding, in the limit, the fixed point of Φ [1]; that is,

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

where "$\circ$" represents function composition.

B. Fractal Block Image Coding

The principle of fractal image coding can be explained in terms of equation (4). Given an image $p \in \Im$, the goal is to find a contractive transformation, Φ, for which p is a fixed point of

Φ, or is sufficiently "close" to the fixed point. If such a Φ can be found, then it may be used as the image code. The construction of Φ is commonly known as the "inverse" problem.

In the Iterated Function System (IFS) proposed by Barnsley [1], the mapping for an image is defined by a set of P contractive transformations acting upon the entire image,

$$W(q) \equiv \sum_{i=1}^{P} w_i(q) \text{ , } q \in \Im, \tag{5}$$

where the set $\{\Im, w_1, w_2, \ldots, w_P\}$ is denoted a hyperbolic IFS. Note that each w_i is applied to an entire image, and it is for this reason that this construction is considered "global". When using (5) to solve the inverse problem, various means have been used in selecting the w_i; these include methods based on the Collage Theorem [1] and the wavelet transform [8].

Fractal block coders, on the other hand, do not apply contractive transformations globally. Rather, FBCs rely upon a single transformation which is mapped to *each* block of a partitioned image. The source of each such transformation is another block derived from the *same* image. This block-wise "self-transformability" results in the original image being encoded in terms of itself; it is for this reason that the term "fractal" is used [3]. This approach to fractal coding has also been formalized in terms of a Partitioned Iterated Function System (PIFS) [7].

Consider the non-overlapping partition of an image, $q \in \Im$, into range blocks of size $B \times B$,

$$q = \bigcup_{i=1}^{N_R} r_i, \quad r_i \cup r_j = \varnothing \text{ for } i \neq j, \ r_i \in \Re, \tag{6}$$

where r_i represents the restriction of the image q to i^{th} range block, $\Re$ denotes the space of discrete domain images of size $B \times B$, and $\varnothing$ denotes no block overlap. We denote $\{r_i\}$ as the *range pool*. The domain pool, $\{d_j\}$, which serves as the basis for coding the range blocks, is comprised of partitioning the same image into a set of possibly overlapping blocks of size $D \times D$,

$$q = \bigcup_{j=1}^{N_D} d_j, \ d_j \in \Delta, \tag{7}$$

where Δ is the space of discrete domain blocks of size $D \times D$.

The function used to map between the domain and range pool is given by

$$\tau_i : \Delta \to \Re \equiv \alpha_i (\iota_{I(i)} (S (d_{N(i)}))) + o_i \ , \tag{8}$$

where $\alpha_i \in R$ is contrast scaling, $o_i \in R$ is an offset, $\iota_{I(i)}$ is a pixel shuffling operation (isometry), $d_{N(i)}$ is the mapped domain block, and $S(\bullet)$ represents a spatial contraction operator which takes a block of $D \times D$ pixels to another block of $B \times B$ pixels [3][7]. The pixel shuffling operation, $\iota_{I(i)}$, is selected from K possible operations which include rotation, reflection and combinations thereof, while the mapped domain block, $d_{N(i)}$, is from the pool of N_D possible domain blocks. The functions $I(i)$ and $N(i)$ provide the mapping from the i^{th} range block to the appropriate isometry or domain block.

The mapping process for each range block consists of determining α_i, o_i, $I(i)$, and $N(i)$ such that resulting block, $\tilde{r}_i = \alpha_i(\iota_{I(i)}(S(d_{N(i)}))) + o_i$, is "close" to the original range block r_i; that is, $d(\tilde{r}_i, r_i)$ is a minimum. Using the distance metric (1) and given two blocks r_i and $\iota_k(S(d_j))$, a closed form expression is available to compute α_i and o_i such that $d(\tilde{r}_i, r_i)$ is minimum [7]. Thus, the process of encoding each r_i is reduced to selecting each block from the set $\{\iota_k(S(\{d_j\}))\}$ (of size KN_D), computing α, o, and evaluating $d(\tilde{r}_i, r_i)$. The k, $1 \le k \le K$, and j, $1 \le j \le N_D$, which result in the minimum overall error are selected as the values for the mapping functions, $I(i)$ and $N(i)$, respectively, with the parameters, α_i, and o_i set to the corresponding α, and o.

With each block transform τ_i determined, the transform for the entire image may be written as

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

If τ is contractive, then (4) may be employed during the decoding. Specifically, the image may be decoded by iteratively applying τ to any point $f_0 \in \Im$, 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 image. (In practice, the number of iterations required for convergence is small - typically f_7 is used as an approximation of f_p.)

It is important to understand the conditions under which τ is contractive. Viewed strictly at a block level, it is necessary that

$$|\alpha_i| < 1 \tag{11}$$

in order for the corresponding τ_i to be contractive when using metric (1). In practice, however, (11) is unnecessarily restrictive [7][6]. In order to understand the reason for this, the interdependence of each τ_i (via d_j^i in (8)) and the iterative nature of the decoding algorithm (via (10)) must be considered.

During the n^{th} decoding iteration, each d_j in (7) is computed from the image approximation, f_{n-1}. Thus, each d_j is dependent upon the action of previously applied τ_i. If a d_j is mapped to a transformation via (8), say to τ_k, then τ_k will also be dependent upon other previously applied mappings. Consequently, even if τ_k is not strictly contractive by (11), it may behave contractively at iteration n if it is dependent upon other sufficiently contractive transformations (applied during the previous $n-1$ iterations). This image specific interdependence makes the exact conditions required for a contractive τ difficult to determine. If τ contains τ_i which are not individually contractive but which are "dominated" by other contractive transformations, τ is termed eventually contractive [7]. In practice, we have always found that $|\alpha_i| < 1.5$ yields an eventually contractive τ.

In order to make each τ_i contractive, it is also unnecessary that a spatial contraction operator, $S(\bullet)$, be included in (8). This is a result of the metric (1) being defined only over pixel values and not upon any spatial properties of the image block. It is, therefore, *possible* that the domain blocks and range blocks be of the same size. *The principal motivation for using a spatial contraction operator, however, is to propagate image detail from one scale to another.* In fact, if a continuous domain image model is used rather than the discrete model described here, it is the spatial contraction operation which allows detail to be provided at sub-pixel resolutions. For a more detailed consideration of this issue refer to [7].

C. Using Range Blocks of Different Sizes - Multilevel Partitioning

As with all block-based coders, FBCs are subject to performance and quality trade-offs when selecting the range block size to use: large range blocks are desirable in smooth image areas and lead to higher compression rates, while smaller blocks are better suited to textural areas and have faster associated computation times (for example, when computing the distance measure (1)). A simple method which combines the advantages of both large and small range blocks is the quadtree approach [7], in which a range block originally of size $B \times B$ is split into four range blocks of size $B/2 \times B/2$ when the error in approximating the large block is beyond a certain threshold, T_0. We denote the partition level as p, $0 \le p \le P$, where P represents the largest number of splits a range block may undergo.

It is important to note that each size of range block requires a different domain pool, and that the smaller the range block the larger the corresponding domain pool. Consequently, as P increases, the length of the search required for smaller blocks will further increase overall encoding time.

III. A Multiresolution Approach to Constraining the Domain Pool Search

In this section we show how multiresolution analysis is related to fractal block-based image coding. We further show that, by extending the multiresolution analysis to more than one resolution level, the domain block search can be constrained by first encoding a lower resolution image and propagating the results to coding at higher resolutions.

A. A Multiresolution View of the Domain Pool

Typically, the spatial contraction operation used in (8) averages over $M \times M$ pixels [3][6][7], where

$$M = \frac{D}{B}, \tag{12}$$

and the averaging operation is given by

$$S(d_j)_{k,l} = \frac{1}{M^2}\left(\sum_{p=0}^{M-1}\sum_{r=0}^{M-1}(d_j)_{kM+p,\,lM+r}\right). \tag{13}$$

The domain blocks, d_j, can be selected from anywhere in the original image, and are usually obtained by taking a $D \times D$ window and sliding it over the image by increments of $\delta_1 \times \delta_2$ pixels [3]. The increments used in FBCs vary from $\delta_1 = \delta_2 = 1$ [7] to $\delta_1 = \delta_2 = D$ [6] (which results in a non-overlapping image partition for the domain pool).

In multiresolution signal analysis, a signal is decomposed into a low resolution approximation signal and a series of difference signals, which provide the detail between one resolution and the next highest [9]. For example, a signal q^0 can be represented by a low resolution signal q^{-1} and a detail signal h^{-1}; that is $q^0 \rightarrow (q^{-1}, h^{-1})$. The process may then be repeated iteratively to the low resolution signal q^{-1}, yielding $q^0 \rightarrow (q^{-2}, h^{-2}, h^{-1})$, $q^0 \rightarrow (q^{-3}, h^{-3}, h^{-2}, h^{-1})$, etc. In this contribution, we restrict our attention to discrete domain signals (i.e. those which represent spatially sampled images), and thus decreases in resolution are made via low pass filtering, detail signals are obtained by high pass filtering, and changes in scale are made by subsampling [12].

Viewing (12) and (13) from a multiresolution perspective [12], it is clear that *FBCs code image blocks by mapping them, via an affine transformation, to lower resolution versions of other image blocks, where the scaling between resolutions is given by* $M \times M$. If, however, we restrict δ_1 and δ_2 to be increments of M, that is,

$$\delta_1 = kM,\ \delta_2 = lM,\ k, l \in Z, \tag{14}$$

then the domain blocks can be selected *directly* from a low resolution image, q^{-1}, obtained from the original image q^0 by filtering and (rectangular) subsampling at the rate $M \times M$. The corresponding increments used in the low resolution image are k and l.

In other words, *by restricting the locations of the domain blocks as in (14), we can extend the multiresolution perspective from the block level to the image level.* In doing so, more sophisticated multiresolution techniques, such as the wavelet transform [9]-[11], can be used when creating the blocks $S(\{d_j\})$. Throughout this contribution, we restrict δ_1 and δ_2 as in (14) when creating the domain pool.

B. Constraining the Domain Pool Search by Exploiting Similarity Invariance at Multiple Resolutions

The mapping between the domain pool and the range pool, $N(i)$, essentially indicates which areas of the image are similar (via some isometry) to each other. *Due to the nature of most practical images, it is reasonable to expect that such similarities also exist at lower resolutions; that is, if two areas of an image are similar at one image resolution, then they would also be similar at a lower resolution.* It is this guiding principle that underlies the MCP algorithm, and is shown here to lead to greater encoding efficiencies.

If the image to be coded, q^0, is decomposed into lower resolution images, q^{-1}, q^{-2}, etc., and coding is first performed on a low resolution image, say q^r, then the mapping obtained at resolution r should provide, using the above principle, an indication of where to search when encoding q^{r+1}. The benefit to this approach is that the corresponding block sizes are smaller at the lower resolutions, and therefore the calculation time for each transformation τ_i is considerably faster.

Notation

> In order to differentiate values of functions between resolutions, we extend the superscript notation used above. For example, $N^r(i)$, is used to indicate the domain block mapping function at the r^{th} resolution level, r_i^r indicates the i^{th} range block at the r^{th} resolution level, etc. Note that a large negative value of r indicates a low resolution image, and that $r = 0$ indicates an image at its original sample rate.

Using this notation, *the goal of the MCP algorithm is to limit the search required to determine $N^r(i)$ by using the mapping $N^{r-1}(i)$. The form of constraint that we propose limits the* *domain pool search to those blocks which originate in a square,* C_i^r*, centred around the domain block in* q^{r-1} *corresponding to* $N^{r-1}(i)$ *The width and height of the constraint square is an integer multiple,* F*, of the size of the range block* r_i^r*. For the MCP algorithm, the constraint square* C_i^r *is a function of the mapping* $N^{r-1}(i)$*.*

The multiresolution constraint area for range block r_i^0 is illustrated in Figure 1. Recall, that since we restrict selection of domain blocks as in (14), the image q^{-2} is the source of domain blocks for encoding q^{-1}, and similarly, q^{-1} is source of domain blocks when encoding q^0.

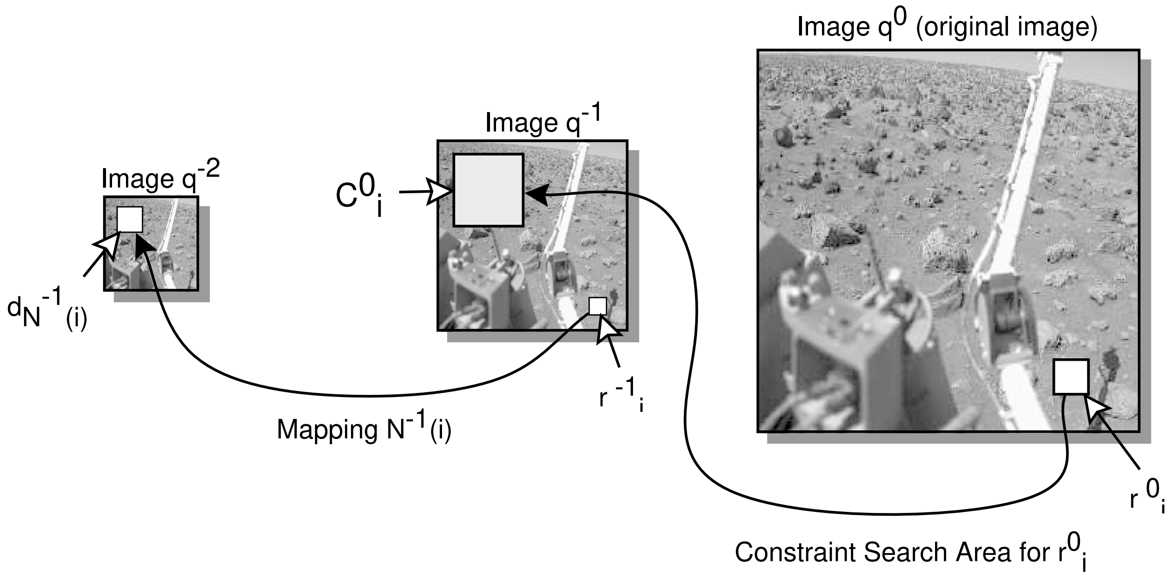


Figure 1
Illustration of Multiresolution Constrained Search For A Range Block at Resolution 0

Clearly, it is necessary that corresponding range blocks at each resolution, r_i^r, r_i^{r+1}, etc., represent the same portion of the image in order to use a constrained search. That is, the range block covering used at one resolution should correspond to the covering at the next resolution. A strict correspondence, however, would not allow the splitting of range blocks according to the multilevel partitioning scheme at any resolution except the lowest. To overcome this difficulty, we permit a range block to be split at any resolution. In the situation where a block is split at resolution r and there is no corresponding block at resolution $r-1$, a full domain block search is used. Furthermore, instead of using a fixed splitting threshold value T for all resolutions, we have found it beneficial to use a threshold value which is a function of resolution, r,

$$T(r) = \frac{T_0}{|r|+1}, \qquad \textbf{(15)}$$

where T_0 is the threshold value used at resolution of the original image ($r = 0$).

A pseudo code description of the MCP algorithm that we have implemented is given below. The variable R represents the lowest resolution level used. Recall that P represents the maximum number of splits a range block may undergo.

Pseudo Code Description of the Multiresolution Constraint Propagation Algorithm

for each resolution $r = R$ *to* $r = 0$

    / create the range pool for this resolution */*

    if $r = R$ *then*

        create the range pool $\{r_i^R\}$ *such that the block sizes are* $\frac{B}{|R|+1} \times \frac{B}{|R|+1}$

    else

        create the range pool $\{r_i^r\}$ *such that it represents the same proportional image covering as* $\{r_i^{r-1}\}$

    / process each range pool block */*

    for each range block r_i^r *in* $\{r_i^r\}$ *do*

        if $r = R$ *or* r_i^r *is a block which has been split at the current resolution then*

            / full search */*

            find the transformation τ_i^r *(and hence mapping* $N^r(i)$ *) by searching all domain blocks from the domain pool* $\{d_j^r\}$

        else

            / try constrained search */*

            find the transformation τ_i^r *(and hence mapping* $N^r(i)$ *) by performing a multiresolution constrained search of* $\{d_j^r\}$*. (Search only domain blocks originating in the area* C_i^r*, where* C_i^r *is based upon* $N^{r-1}(i)$ *)*

            / if error found by constrained search is too large, try a full search */*

            if $d(r_i^r, \tilde{r}_i^r) > T(r)$ *and* $r \neq 0$ *then*

                find the transformation τ_i^r *(and hence mapping* $N^r(i)$ *) by searching all domain blocks from the domain pool* $\{d_j^r\}$

        endif

        / split the block or accept transformation? */*

        if $d(r_i^r, \tilde{r}_i^r) > T(r)$ *and the partition level of* r_i^r *is less than or equal to P then*

            split r_i^r *into 4 blocks, increment their partition level, and add to the range pool* $\{r_i^r\}$

        else

            accept transformation τ_i^r *for block* r_i^r

    endfor / each range block */*

endfor / resolution */*

C. Analysis of Performance

The MCP algorithm utilizes many parameters, several of which directly effect encoding performance. In this section, we investigate the computational performance the MCP algorithm with respect to these critical parameters.

Regardless of the search process, the selection of $N^r(i)$ requires that the variables α, o, and the isometry mapping $I(i)$ be determined for each domain block tested. The computation of α and o depend upon the size of the block being encoded, while the search for $I(i)$ is dependent upon the number of isometries, K. Since K does not change between resolutions and partition levels, we omit this factor and write the computation time for each test as being of the order $O(n^2)$, where n is the row size or column size (since square blocks are used) of the range block being encoded and is given by

$$n = \frac{B}{(|r|+1)\ (p+1)} \ . \qquad \textbf{(16)}$$

Therefore, the computation time for each domain candidate decreases approximately as the square of the resolution or partition level.

Note that there are at least as many range blocks to be encoded at each resolution as there are at the highest resolution, and that the number of blocks in the domain pool is independent of resolution (though not partition level). Consequently, the total number of searches required in the MCP technique is larger than in the full search method. *The performance benefit in the proposed MCP technique arises from performing short constrained searches on the larger blocks of the higher resolution images, while restricting full searches to the smaller blocks at the lowest resolution images, range blocks which have just been split, or to range blocks for which the constrained search fails to provide a suitable domain block* (refer to pseudo code algorithm for details).

As stated earlier, the number of domain blocks is independent of resolution but not of partition level. We use the function $N_D(p)$ to denote the number of domain blocks at a partition level. Note that $N_D(p)$ is dependent upon the increments δ_1 and δ_2. The function $\gamma(p, r)$, $0 \le \gamma(p, r) \le 1$, is used to denote the proportion of domain blocks searched at partition level p and resolution r. Clearly, $\gamma(p, r)$ is dependent upon the threshold value T_0, image q^0 and constraint area factor F, and thus must be obtained experimentally. Note, however, that for the MCP algorithm, $\gamma(p, R) = 1$, where R is the lowest resolution used. We further use the function $N_S(p, r)$ to represent the number of range block searches performed at a given resolution and partition level. Note that $N_S(p, r)$ is dependent upon the type of search technique used. To distinguish $N_S(p, r)$ between search techniques, an additional superscript indicating the search method is used.

Given these definitions, the computational advantage, η^{MCP}, in using the proposed MCP algorithm versus the full search algorithm is defined by

$$\eta^{MCP} = \left(\sum_{p=0}^{P} N_D(p) \sum_{r=R}^{0} N_S^{MCP}(p, r)\,\gamma(p, r)\,O(n^2)\right) \Big/ \left(\sum_{p=0}^{P} N_D(p)\,N_S^{Full}(p, 0)\,O(B^2)\right). \quad \textbf{(17)}$$

Equation (17) indicates that, in order for the MCP algorithm to be advantageous, it is necessary that $\gamma(p, r) \ll 1$ for $r \neq R$. Experimentally obtained values for $\gamma(p, r)$ and $N_D(p)$, provided in Section V, indicate that this is indeed the case for the selected parameters. Further discussion of the performance of this algorithm is made in SectionV.

IV. Constraining the Domain Pool Search Using a Randomly Selected Area

When considering the above MCP algorithm, it is natural to ask how effective and useful are the search area constraints. Could, for example, a random selection of a same-sized search area be as effective, and thus preclude use of the complexities involved in the MCP algorithm? In this section, we present an algorithm based upon using a randomly selected search area and show that this is indeed the case.

A. Random Constraint Search Algorithm

In the MCP algorithm, the constraint area for r_i^r, C_i^r, is a function of domain block mapping, $N^{r-1}(i)$, obtained for r_i^{r-1}. For the proposed RCS algorithm, however, the *constraint search area* C_i^r *is a function of a randomly selected domain block from* q^{r-1}. Since no information is required from any resolutions other than q^0 and q^{-1} (which is the source of the domain pool), *the random constraint algorithm has the benefit that only the image at the original resolution image needs to be encoded.*

The pseudo code description of this algorithm is presented below. Note that in order to provide tolerance against selecting a very poor constraint area, we allow several (*MaxTries*) constrained searches per range block and select the best corresponding domain block from those searches. Also note that the “full search” fractal coding algorithm can be obtained from this pseudo code by setting $MaxTries = 1$, and performing a full search of the domain pool rather than a random one.

Pseudo Code Description of Random Constraint Search Algorithm

/ create the range pool */*
initialize the range pool $\{r_i^0\}$ *such that the block sizes are* $B \times B$

/ process each range pool block */*
for each range block r_i^0 *in* $\{r_i^0\}$ *do*
    besttransform = NULL; besterror = infinity;
    tryagain = TRUE; try = 1;
    while (tryagain is TRUE and try <= MaxTries) do
        / use random constraint search */*
        find the transformation τ_i^0 *(and hence mapping* $N^0(i)$ *) by performing a constrained search of* $\{d_j^0\}$. *(Search only domain blocks originating in the area* C_i^0, *where* C_i^0 *is based upon a randomly selected location of* q^{-1} *)*

        / keep best try so far */*
        if $d(r_i^0, \tilde{r}_i^0) <$ *besterror then*
            besttransform = τ_i^0*;*
            besterror = $d(r_i^0, \tilde{r}_i^0)$*;*

        / if error below threshold, don't search any more */*
        if besterror <= T_0 *then*
            tryagain = FALSE;

        increment(try);
    endwhile

  / split the block or accept transformation? */*
  if besterror > T_0 *and the partition level of* r_i^0 *is less than or equal to P then*
    split r_i^0 *into 4 blocks, increment their partition level, and add to the range pool* $\{r_i^0\}$
  else
    accept besttransform as the transformation for block r_i^0

endfor / each range block */*

B. Analysis of Performance

The performance analysis of the RCS algorithm is simpler than for the MCP algorithm. Using the function $\gamma(p, 0)$ to represent the proportion of domain blocks searched at partition level p, the computational advantage in using the RCS algorithm versus the full search algorithm is given by

$$\eta^{RCS} = \left(\sum_{p=0}^{P} \gamma(p, 0)\, N_D(p)\, N_S^{RCS}(p, 0)\, O(B^2)\right) \Big/ \left(\sum_{p=0}^{P} N_D(p)\, N_S^{Full}(p, 0)\, O(B^2)\right) \quad \textbf{. (18)}$$

As before, the function $\gamma(p, 0)$ depends upon several parameters as well as the image being coded. Values of $\gamma(p, 0)$ obtained from test images are presented in Section V.

V. Results and Discussion

In this section, results are presented comparing the full search FBC and proposed constrained search algorithms. Experimentally measured values for the functions $\gamma(p, r)$, $N_S(p, r)$, and $N_D(p)$ are also provided.

A. Choice of Parameters

It is clear that there are a large number of parameter choices to be made for a FBC. Table 2 summarizes the parameter values used in the full search, MCP and RCS algorithms.

Table 1 . Parameter Values

Parameter	Value	Remarks
B	8	Range Blocks originally of size 8x8 (Before Quadtree Splitting)
D	16	16x16 Domain Blocks (thus M = 2)
P	1	Two Level Partition (8x8 and 4x4 Range Blocks are allowed)
T_0	15.0	Block Splitting Threshold at Resolution 0
$\delta_1 = \delta_2$	4	Results in Domain Block Overlapping (2x2 separation of blocks in *low resolution image*)
$\lvert\alpha\rvert$	1.5	Yields Eventually Contractive τ.
α_B	6	Bits used to store scaling α
o	-255 to 255	Offset Value
o_B	8	Bits Used to store offset o
ι_B	3	Bits Used to represent isometries (8 isometries used as per [2]).
$S(\bullet)$	Equation (13)	Pixel Averaging
R	-2	Multiresolution coding algorithm starts at q^{-2}
F	2	Constraint Area Spreading Factor
MaxTries	4	Maximum of 4 Random Searches Per Range Block
f_p	f_7	7th iteration of decoding algorithm used as decoded image

B. Computation of Bit Rates

In computing the bit rate, it is assumed that the range blocks are stored in a fixed order so that no bits are required to encode range block positional information. Furthermore, because we allow a P level partition, P_B bits must be allocated to identify the size of the range block; for the value $P = 1$ used here, $P_B = 1$ bit is required. The isometries used are the same as described in [3], and thus 3 bits are required to encode the isometry ($\iota_B = 3$).

Using the above parameter values and assuming the original image is of size $N_1 \times N_2$ the bit rate of the FBC can be computed as

$$\frac{N_R (P_B + o_B + \alpha_B + \iota_B + D_{row} + D_{col})}{N_1 N_2} \frac{\text{bits}}{\text{pixel}}, \qquad \textbf{(19)}$$

where D_{row} and D_{col} are the number of bits required to encode the position of domain blocks *in the low resolution image*, and N_R is the number of range blocks. Note that D_{row} is a function of δ_1 and N_1, while D_{col} is a function of δ_2 and N_2. Using the parameter values provided in Table 2, $D_{row} = D_{col} = 6$ bits, and thus each range block requires a total of 30 bits to encode.

In addition to computing the bit rate, we measure image distortion using the peak-to-peak signal to noise ratio (PSNR) between two images q and $\tilde{q}$, given by

$$\text{PSNR} = 10\log_{10}\left(\frac{\max(\text{q}) - \min(\text{q})}{MSE(q, \tilde{q})}\right), \qquad \textbf{(20)}$$

where $MSE(q, \tilde{q})$ is the mean squared error between q and $\tilde{q}$.

C. Results Using Test Images

The full search, MCP and RCS algorithms have been used to code the standard 256×256, $q_{bpp} = 8$ bits/pixel, “Lenna” image. The original image, as well as the decoded images are shown in Figure 3, with the corresponding performance results provided in Table 2. Results for the “Lander” image are presented in Figure 5, with the corresponding performance results given in Table 2. Encoding times are normalized to the full search algorithm. Actual times have been taken from the overall user time take for the programs to complete.

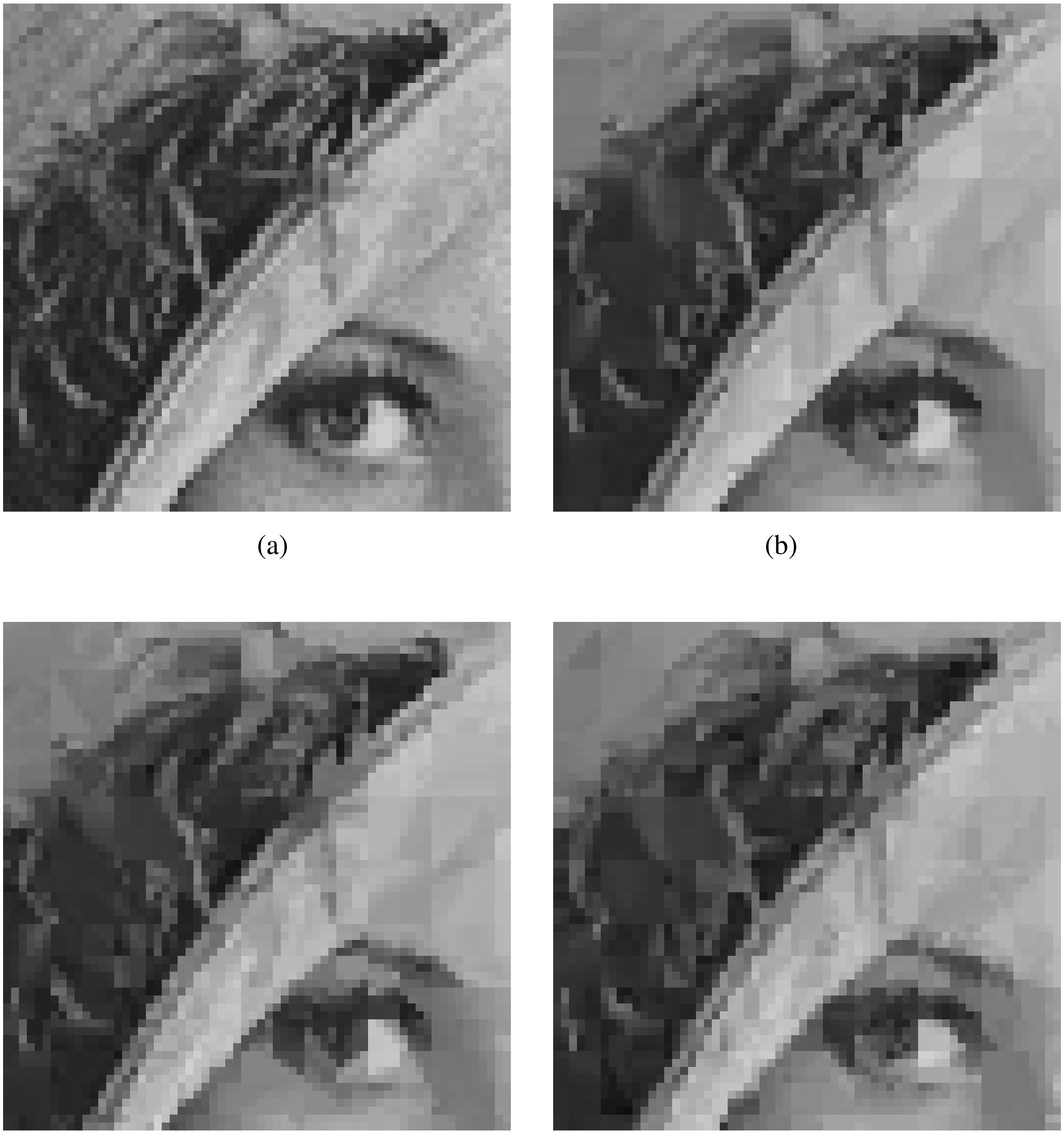
(a)

(b)

(c)

(d)

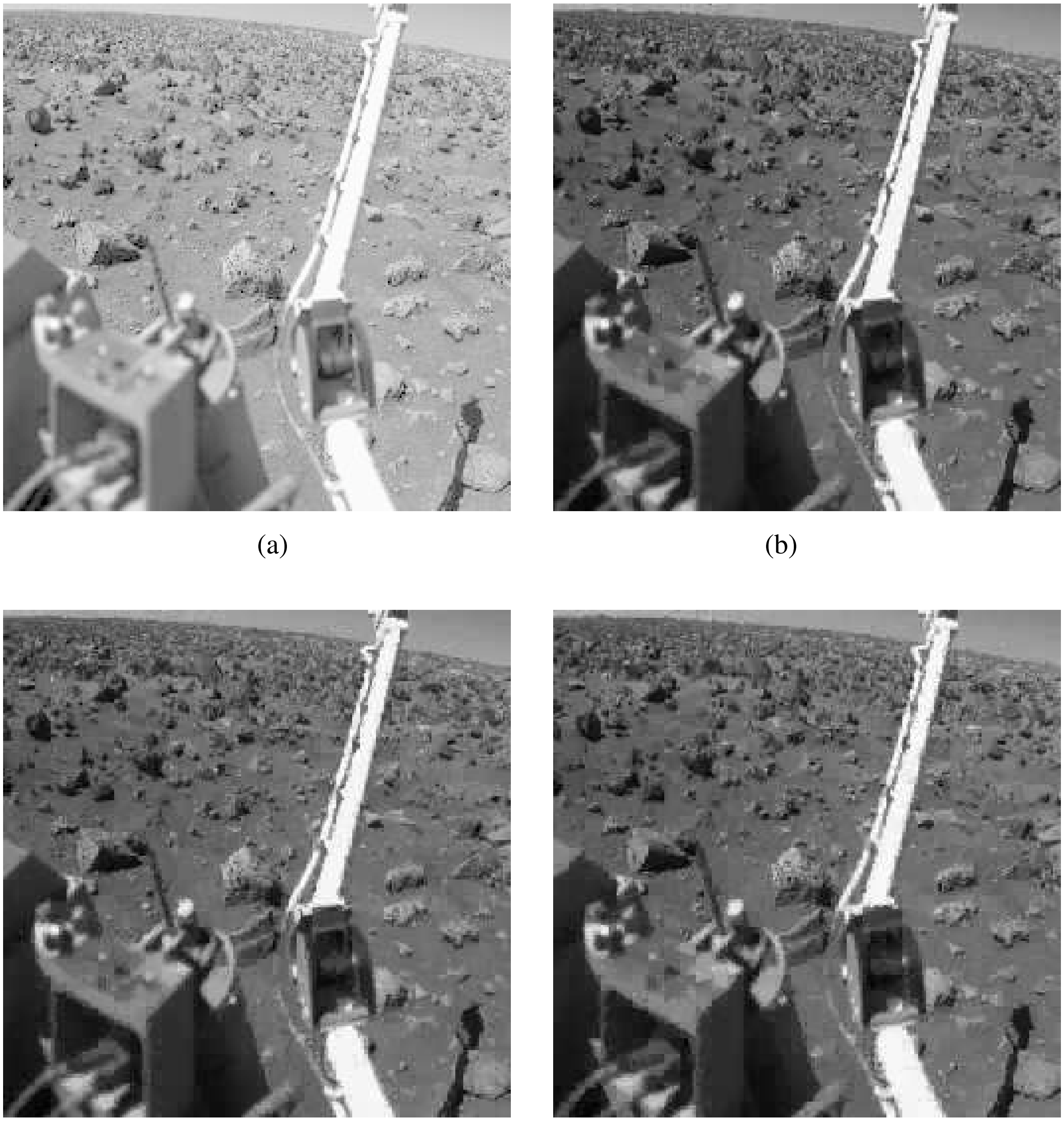
Figure 2
Coding Results for “Lenna” - (a) Original Image (b) Full Search Algorithm (c) MCP Algorithm (d) RCS Algorithm



(a) (b)

(c) (d)

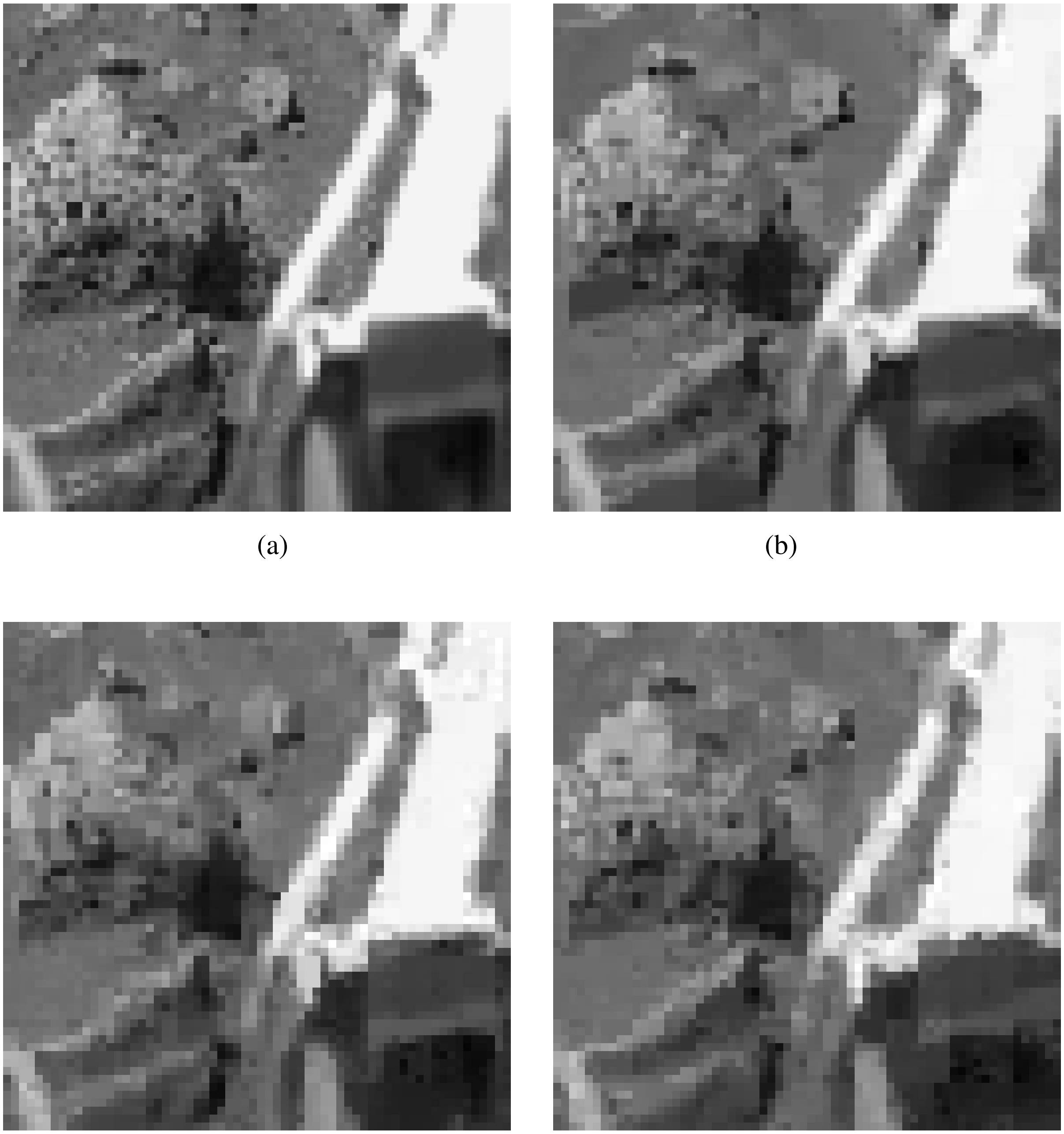
Figure 3
Coding Results for "Lenna" - Detail Images - (a) Original Image (b) Full Search Algorithm (c) MCP Algorithm (d) RCS Algorithm



(a) (b)

(c) (d)

Figure 4
Coding Results for “Lander” - (a) Original Image (b) Full Search Algorithm (c) MCP Algorithm (d) RCS Algorithm



(a) (b)

(c) (d)

Figure 5
Coding Results for "Lander" - Detail Images - (a) Original Image (b) Full Search Algorithm (c) MCP Algorithm (d) RCS Algorithm

Table 2 . Coding Performance Results Corresponding To Figure 3 and Figure 5

Coding Method	PSNR (dB)	Bits Per Pixel	En-coding Time[a]	Total Number of Range Blocks	Constraint Search Factor $\gamma(p, r)$	Number of Searches $N_S(p, r)$	Length of Domain Block Search $N_D(p)$ [b]
			"Lenna" Image				
Original Image (Figure 2 (a))	—	8	——	——	————	——	——
Full Search (Figure 2 (b))	26.99	0.727	1.0	1588	————	$N_S(0, 0) = 1024$ $N_S(1, 0) = 752$	
Two Level Multi-resolution Constrained Search (Figure 2 (c))	26.27	0.813	0.77	1777	$\gamma(0, -2) = 1.0$ $\gamma(1, -2) = 0.0$ $\gamma(0, -1) = 0.286$ $\gamma(1, -1) = 1.0$ $\gamma(0, 0) = 0.074$ $\gamma(1, 0) = 0.117$	$N_S(0, -2) = 1024$ $N_S(1, -2) = 0$ $N_S(0, -1) = 1328$ $N_S(1, -1) = 904$ $N_S(0, 0) = 798$ $N_S(1, 0) = 1004$	$N_D(0) = 961$ $N_D(1) = 3969$
Random Search (Figure 2 (d))	25.54	0.785	0.10	1714	$\gamma(0, 0) = 0.601$ $\gamma(1, 0) = 0.017$	$N_S(0, 0) = 1791$ $N_S(1, 0) = 1947$	
			"Lander" Image				
Original Image (Figure 4 (a))	—	8	——	——	————	——	——
Full Search (Figure 4 (b))	27.09	1.146	1.0	2503	————	$N_S(0, 0) = 1024$ $N_S(1, 0) = 1972$	
Two Level Multi-resolution Constrained Search (Figure 4 (c))	26.10	1.263	0.74	2758	$\gamma(0, -2) = 1.0$ $\gamma(1, -2) = 0.0$ $\gamma(0, -1) = 0.428$ $\gamma(1, -1) = 1.0$ $\gamma(0, 0) = 0.071$ $\gamma(1, 0) = 0.143$	$N_S(0, -2) = 1024$ $N_S(1, -2) = 0$ $N_S(0, -1) = 1661$ $N_S(1, -1) = 2020$ $N_S(0, 0) = 965$ $N_S(1, 0) = 2312$	$N_D(0) = 961$ $N_D(1) = 3969$
Random Search (Figure 4 (d))	25.66	1.201	0.10	2623	$\gamma(0, 0) = 0.601$ $\gamma(1, 0) = 0.017$	$N_S(0, 0) = 2708$ $N_S(1, 0) = 4288$	

a. Encoding times are normalized to the Full Search method for the corresponding image. The full search algorithm for "Lenna" required 1Hr 41min to complete on a 25MHz NeXTStation, while the "Lander" image required 3Hr 17min to complete.

b. $N_D(p)$ is independent of resolution except at $r = -2$, where no split is permitted since the resulting blocks would be of size 1x1.

D. Comparative Interpretations of Data

For the tested images, the MCP algorithm reduces encoding time by approximately 24%, while the RCS algorithm reduces encoding time by 90%. Both methods, however, yield slightly poorer performance than the full search algorithm. In particular, both proposed methods require a higher number of bits per pixel and have a lower PSNR value. When comparing the MCP and RCS algorithms, the MCP algorithm provides better results. It is important to remember, however, that the MCP algorithm is considerably more complicated. Overall, both proposed methods trade off performance quality for increased encoding speed.

In terms of the subjective quality of the images, there is little difference between the full search and MCP techniques. The RCS algorithm for the "Lenna" image, however, does show increased signs of blockiness, particularly along some edge boundaries and in textural areas such as the feathers around the hat. In the "Lander" image, the blockiness is more apparent in and around the lander equipment, but does significantly degrade the subjective quality in the textural areas of the image. Clearly, the RCS algorithm would be beneficial to use in images that contain large textural areas which are predominately "background".

Using equation (17) in conjunction with the results for the "Lenna" image from Table 2, the computational advantage of the MCP algorithm, relative to that of the full search algorithm, is given by

$$\eta^{MCP} = \frac{984\times10^{3}O(4) + 3.95\times10^{6}O(16) + 762\times10^{3}O(64)}{2.98\times10^{6}O(16) + 984\times10^{3}O(64)}. \qquad \textbf{(21)}$$

Equation (21) illustrates that the MCP algorithm achieves a performance gain by shifting a significant portion of the search effort to the medium sized blocks (i.e. 4×4). There are, however, still a large number of searches for the highest resolution blocks, indicating that speed of the algorithm may be further reduced by changing the threshold function (15), or by entirely restricting the search when encoding the highest resolution image.

Using equation (18), the computational advantage of the RCS algorithm, relative to that of the full search algorithm, is given by

$$\eta^{RCS} = \frac{129\times10^{3}O(16) + 103\times10^{3}O(64)}{2.98\times10^{6}O(16) + 984\times10^{3}O(64)}. \qquad \textbf{(22)}$$

The performance gain of the RCS algorithm, given in equation (22), illustrates that searches at both partition levels are reduced from the full search algorithm. It is interesting to note that there is a larger proportional decrease in blocks at the first partition level, again indicating that the algorithm is sensitive to the threshold function, given in equation (15).

Overall, these results illustrate that the proposed constraint mechanisms do reduce encoding time while maintaining an acceptable level of image quality. These algorithms, however, contain many variables and further investigation and study of their effects on image quality and compression rates is warranted.

VI. Conclusion and Directions for Future Research

The multiresolution view of the domain pool has been shown to allow techniques other than simple pixel averaging to be used when creating the domain pool source image. Although not reported here, we have made preliminary attempts using the low resolution approximation signal from the 2-D discrete wavelet transform (*2-D DWT*) to create the domain pool source image. We will also be investigating the use of the *2-D DWT* difference images, which contain directionally sensitive edge information, as the domain pool source. Results from these inquiries will be presented in the future.

Another direction that needs to be pursued is the application of the constraint search technique to video coding, implying extension to three dimensions. An advantage in using a constrained search technique is that the entire domain pool does not have to be available to perform each search. For 2-D images, the number of domain blocks can be made sufficiently small so that having all of them available at one time is not a problem (even if they do not all get tested). In extending the algorithm to 3-D, however, the number of domain blocks can become prohibitively large and thus the use of a constrained search is necessary. Given the speed improvement in using the random search method, we envisage a video coding system based upon short temporal (e.g. 10 frame) sequences in which a random search of preceding and succeeding frames is performed.

In this paper, we have presented two methods to constrain the search space for fractal block-based coders (FBCs). We have first described the theory and implementation of a full search block based coder and shown that, essentially, FBCs encode image blocks by mapping them to lower resolution versions of the other image blocks. With this insight, we have proposed a constraint technique based upon a multiresolution decomposition of the original image. This multiresolution based method reduces encoding time not by decreasing the overall number of searches, but rather by performing full searches for lower resolution blocks and constraint based searches on higher resolution blocks. The constraint itself was based upon the idea that image areas which are similar retain their similarity at different image resolutions. In the second method we employ, domain blocks whose source is from a randomly selected areas of the low resolution domain block source image. The results of encoding two different images have been described in detail and show that the proposed techniques significantly reduce encoding time at the expense of a small corresponding decrease in the coding bit rate and image quality.

VII. References

[1] M. Barnsley, *Fractals Everywhere*, Academic Press, 1988.

[2] A. Jacquin, *A Fractal Theory of Iterated Markov Operators with Applications to Digital Image Coding*, Ph.D. dissertation, Georgia Inst. of Technology, 1989.

[3] ———, “Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations,” *IEEE Trans. Image Processing,* vol. 1, no. 1, pp. 18-30, January 1992.

[4] ———, “A Novel Fractal Block-Coding Technique for Digital Images,” *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Albuquerque, pp. 2225 - 2228, April 1990.

[5] ———, “Fractal image coding based on a theory of iterated contractive image transformations,” SPIE, Vol. 1360, Visual Communications and Image Processing 1990, pp. 227 - 239.

[6] G. ∅ien, 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.

[7] Y. Fisher, “Fractal Image Compression,” SIGGRAPH 1992 Course Notes.

[8] R. Rinaldo, A. Sakhor, “Inverse Problem for Two-Dimensional Fractal Sets Using the Wavelet Transform and the Moment Method,” *IEEE International Conference on Acoustics, Speech, and Signal Processing*, San Francisco, pp. 665 - 668, 1992.

[9] S. Mallat, “A Theory for Multiresolution Signal Decomposition: The Wavelet Representation,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-11, no. 7, pp. 674-693, July 1989.

[10] I. Daubechies, “Orthonormal Bases of Compactly Supported Wavelets,” *Commun. Pure Appl. Math.*, vol. 41, pp. 909-996, Nov. 1988.

[11] I. Daubechies, *Ten Lectures on Wavelets*. Philadelphia: SIAM, CMBS Series, 1992.

[12] O. Rioul and M. Vetterli, “Wavelets and Signal Processing,” *IEEE Signal Proc. Mag.*, October 1991, pp. 14-38.