HIERARCHICAL FAST DECODING OF FRACTAL IMAGE REPRESENTATION USING QUADTREE PARTITIONING

I Sutskover and D Malah *

Technion - I.I.T, Israel

ABSTRACT

Fractal image coding using either fixed-size blocks or variable-size blocks in a quadtree structure is by now common. However, a fast decoding algorithm based on a hierarchical representation, which requires only a finite number of iterations to exactly reach the fixed point of the fractal transformation, has been shown only for fixed-size blocks. In this work, a generalization of the fast decoding algorithm is made, enabling its use with a quadtree image partitioning. A theorem extending the hierarchical representation of the fractal transformation fixed point to include quadtree partitioning is given and proved.

1 INTRODUCTION

A fast decoding method using hierarchical decoding was originally introduced by ϕien *et al* [1] and Baharav *et al* [2], and is reviewed in [3]. Using hierarchical decoding, one can decode an image, i.e., reach the fixed point of the fractal transformation, in a finite predetermined number of iterations.

This fast decoding algorithm was presented only for fixed size range blocks, and hence its use in decoding a quadtree partitioned image was not reported so far. Since quadtree partitioning is a common way to improve the performance of fractal image coders [4], it is of great interest to extend the fast decoding method of [1, 2] to include this partitioning as well.

The concepts of the original fast decoding method are briefly reviewed in section 2. The extended algorithm, and a proof of the generalization is given in section 3. Section 4 demonstrates simulation results of the extended algorithm and section 5 concludes and discusses the applications arising from this work.

2 BLOCK-BASED FRACTAL CODING AND A FAST DECODING ALGORITHM

In [5, 6] Jacquin introduced a practical block-based fractal image coding scheme. Encoding is done by partitioning the image into non-overlapping blocks, called *range blocks* (denoted $\mathbf{R}_i$, where i is the block index), and into typically overlapping blocks twice as large in each coordinate - creating what is called the *domain pool*. A spatial contraction is performed on the domain pool, by an operator φ (usually averaging), so that the resulting blocks (which we denote $\mathbf{D}_j$) are of the size of the range blocks.

*This work was supported by the Ministry of Science, Grant No. 8635196.

Each range block is linearly estimated from a contracted domain block according to $\hat{\mathbf{R}}_i = a_i \cdot \mathbf{D}_{j_i} + \mathbf{b}_i$, where a_i is called the scale parameter, b_i - the offset parameter, and j_i is the index denoting the best domain block to get $\hat{\mathbf{R}}_i$ as close as possible to $\mathbf{R}_i$. Decoding can be done by starting with some initial image, estimating the range blocks of a new image from the contracted domain blocks, and iterating until a fixed point is reached, or nearly so (thus this method is called IFS - Iterated Function System). If the range block estimators are good enough, then the fixed point image is much like the original picture. It should be noted that this method does not really dictate the shape of the range blocks, thus allowing quadtree partitioning, for example.

In [1] a method for enriching the domain pool by a span of vectors is suggested. A simple version of this method, called *DC orthogonalization*, utilizes flat (DC) planes, thus estimating range blocks by:

$$\hat{\mathbf{R}}_i = a_i \cdot \left\{ \mathbf{D}_{j_i} - \bar{\mathbf{D}}_{j_i} \right\} + \bar{\mathbf{R}}_i \qquad (1)$$

where the bar indicates a block of constant value which is the mean of the corresponding block. Using DC orthogonalization requires the triplets $\{a_i, \bar{\mathbf{R}}_i, j_i\}$ to be transmitted/saved as the compressed code, instead of the triplets $\{a_i, b_i, j_i\}$ which where used by Jacquin's code. The amount of information in the code is the same in both methods.

In [2] Baharav *et al* introduced the hierarchical representation and interpretation of the fixed point. Before reviewing their results, several definitions are necessary:

Definition 1 *An image at resolution m,* $\mathbf{Y}_{\frac{1}{2^m}}$*, is the image obtained from an image* $\mathbf{Y}$ *(or* $\mathbf{Y}_1$*) by* m *repeated contractions using the spatial contraction operator* φ*:*

$$\mathbf{Y}_{\frac{1}{2^m}} = \varphi\left\{\mathbf{Y}_{\frac{1}{2^{m-1}}}\right\} = \varphi^{(m)}\left\{\mathbf{Y}_1\right\} \;;\; m \geq 1 \qquad (2)$$

Definition 2 A local transformation t_i *is a set* $\{a_i, r_i, j_i\}$ *used for estimating range block* $\mathbf{R}_i$*.*

Definition 3 *A transformation* $T_{\frac{1}{2^m}}$ *is a union* $\bigcup_i t_i$ *which operates via its local transformations on an image having the size of* $\mathbf{Y}_{\frac{1}{2^m}}$*. The fixed point of this transformation is denoted by* $\mathbf{Y}_f\{T_{\frac{1}{2^m}}\}$*.*

In the sequel, the contracting operator φ will be considered to be an averaging operator, but other linear operators with a mapping property of $2 \times 2 \mapsto 1 \times 1$ that are *commutative with the averaging operations* are valid.

The main theorem of [2, 3] states that the different resolution levels of a **fixed point** $\mathbf{X}_1 = \mathbf{X}_f\{T_1\}$ (which are

$\mathbf{X}_{\frac{1}{2^m}}$ for the different values of m) are the corresponding fixed points of the different transformations $T_{\frac{1}{2^m}}$ (i.e. $X_f\left\{T_{\frac{1}{2^m}}\right\}$ for those m values). This theorem is used to develop a fast decoding algorithm [3] that reaches the fixed point of the transformation in a fixed number of steps that is $\log_2$ of the side length of the square range block.

Two operators are now defined. The first is the zoom-out operator (to be referenced as IZO for IFS Zoom Out): $\mathbf{X}_{\frac{1}{2^{m+1}}} = \varphi\left\{\mathbf{X}_{\frac{1}{2^m}}\right\}$. Contracting a fixed point yields the fixed point at the next lower resolution. The second is the IFS Zoom-In (IZI) operator which is derived directly from the definition of the range block estimator, and is described by:

$$\mathbf{X}_{\frac{1}{2^m}} = \bigcup_i \mathbf{R}_i \; ; \; \mathbf{R}_i = a_i \cdot \left\{\mathbf{D}_{j_i} - \bar{\mathbf{D}}_{j_i}\right\} + \mathbf{r_i} \qquad (3)$$

where $\mathbf{D}_{j_i}$ is of the size of $\mathbf{R}_i$, and is taken from the picture $\mathbf{X}_{\frac{1}{2^{m+1}}}$ with no further contraction (since $\mathbf{X}_{\frac{1}{2^{m+1}}}$ is already a contracted version of $\mathbf{X}_{\frac{1}{2^m}}$). $\mathbf{r_i}$ denotes the value of $\bar{\mathbf{R}}_\mathbf{i}$ in (1) stored in the code. Applying the IZO operator M times to the fixed point picture $\mathbf{X}_1$, where $M = \log_2 B$, results in a picture $\mathbf{X}_{\frac{1}{2^M}}$ in which every range block is one pixel in size, and its value is given explicitly in the code. Hence M IZI operations, starting with $\mathbf{X}_{\frac{1}{2^M}}$, results in the desired image $\mathbf{X}_1$.

3 FAST DECODING USING QUADTREE PARTITIONING

When encoding an image, the smaller the size of the range blocks, the better the quality of the reconstructed image, but the lower the compression ratio. Quadtree partitioning provides a way for improving coder performance by allowing the use of blocks of different sizes depending on the matching quality of corresponding range and contracted domain blocks. Details about the use of quadtrees in fractal coding can be found in [4, Ch. 3].

Encoding with a quadtree, either Jacquin's way or ϕien's way (i.e., using $\{a_i, b_i, j_i\}$ or $\{a_i, r_i, j_i\}$), is done exactly the same as with fixed size range blocks. The fast decoding algorithm of [3] requires however all the range blocks in the picture to be squares of the same size, which is obviously not the case when quadtree partitioning is used. To stress the point: Fast decoding requires $\log_2 B$ iterations, where B is the size of the range block, but which size of block should be used here when we have range blocks of different sizes?!

An extension of the fast decoding algorithm, to include quadtree partitioning of the image, is now presented. This leads to a generalization of the Theorem in [2, 3].

Let the biggest range block in the quadtree-partitioned picture be of size $2^{L_{max}} \times 2^{L_{max}}$, and the smallest one of size $2^{L_{min}} \times 2^{L_{min}}$. Then, according to the fast decoding algorithm above, L_{max} IZI operations should be applied to reconstruct the biggest blocks, while only L_{min} IZI operations should be applied to reconstruct the smallest ones. Let us assume that only blocks of the biggest size are to be used to cover the image in spite of the quadtree partitioning, then an initial image can be constructed from the average values of these blocks. If a block is partitioned by quadtree to its sons, its average is not contained explicitly in the transformation. However, it can be recovered by averaging the averages of its sons. The same action may be taken when one or more of its sons are further partitioned.

This leads us to suggest the following extended fast decoding algorithm:

1. Create an image of the size of $\mathbf{X}_{\frac{1}{2^{L_{max}}}}$. If a pixel in this image corresponds to a range block in the original image (which is of the biggest size), then initiate it with r_i given in the transformation T. Else, this pixel is a contraction of a block in the original image, which is of the biggest size, but is not a range block, and may be considered as a union of smaller range blocks. This pixel is initiated with the weighted average of this union, according to the size of the range blocks in the union, and the exact average of the big block is recovered. This can be done naturally in a recursive way. Call this image *the source image.*

2. Allocate space for an image twice the size, in each coordinate, of the source image. Call this image *the destination image.*

3. For every block in the source image that is a contraction of a range block, use the IZI operator and put the results in the proper place at the destination image. To any pixel in the source image which is not a contraction of a range block, correspond four pixels in the destination image, which should be each a contraction of a range block, or a contraction of a union of range blocks, and therefore these four pixels are set in the same manner as in stage 1.

4. Rename the destination image as source image, and return to stage 2. Repeat stages 2-4 L_{max} times.

5. After the last iteration, the image called *source* is exactly the fixed point of the transformation.

It should be noticed that each range block is built using a number of iterations which is exactly $\log_2$ of its size, thus reaching eventually its real size.

A demonstration of the process is depicted in Fig. 1.

A formal proof will now be given to support this algorithm, claiming that the fixed point can be obtained using a fast hierarchical decoding, even when quadtree partitioning is used.

We'll say that a block (not necessarily a range block) is *limited to resolution n*, if it belongs to the original image $\mathbf{X}_1$, and its size is $2^n \times 2^n$. A block R like this may be contracted n times at most, to remain represented on a discrete grid. The last contraction results in a single pixel (1×1) having the value $\bar{\mathbf{R}}_i$, i.e. the average of the block.

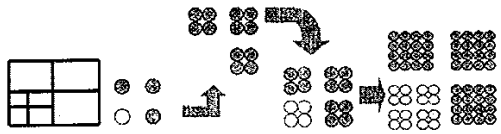

Figure 1: **On the left side, the original quadtree partition. On the right side, the decoding process: all 1×1 blocks (pixels) are initiated using the average values given by the transformation (explicitly or implicitly). Then, IZI operation is performed to get the three 2×2 blocks according to the quadtree structure, and then, the averages of the smaller blocks are implanted. Finally, an IZI operation is performed, this time yielding the whole 8×8 image. More IZI operations may be needed to get the real size of the image.**

Since range blocks of different sizes are present when a quadtree is used, there are range blocks (the smaller ones) which may not be represented by their contracted version after several contractions of the image. This implies that the original transformation T_1 cannot be applied to this image, and therefore, a different version of T must be defined. This is done in the next definition, which replaces definition 3.

Definition 4 *A transformation $T_{\frac{1}{2^m}}$ is a transformation that:*

1. *Operates on an image of the size of $\mathbf{X}_{\frac{1}{2^m}}$.*
2. *Inherits from the transformation $T_{\frac{1}{2^{m-1}}}$ its local transformations $t_i=\{a_i,r_i,j_i\}$ such that:*
   (a) *If t_i relates to a block limited to resolution k and $k\geq m$, then t_i belongs to $T_{\frac{1}{2^m}}$ too.*
   (b) *If t_i relates to a block limited to resolution $m-1$, then this block is represented at the current level by only one pixel, and due to the structure of the quadtree partition, three more blocks like this (limited to resolution m-1) are its neighbors. Thus, instead of the four corresponding local transformations, a new local transformation is defined for $T_{\frac{1}{2^m}}$ by $t_i=\{0,r_i,0\}$ so that the new value r_i (the average of the corresponding block) is equal to the average of these four blocks. Note, that a_i and j_i are irrelevant here, and hence were formally set to 0.*

Theorem 1 Hierarchical fixed point representation (generalization).
Given a transformation $T=T_1=\bigcup_i t_i$ where:

- *$\{t_i\}$ are local transformations which relate to range blocks according to a given quadtree structure.*
- *Range blocks are of size $2^k\times2^k$ $(0\leq k\leq L$, k and L are positive integers). L relates to the size of the biggest range block that exists in the partition.*
- *For every B such that a $B\times B$ range block exists, there is a pool of contracted domain blocks, and the displacement in this pool between every two blocks (in any direction) is an integer multiple of $\frac{B}{2}$.*
- *$\mathbf{X}_1$ is the fixed point of T_1.*

Then

$$\mathbf{X}_f\left\{T_{\frac{1}{2^m}}\right\}=\mathbf{X}_{\frac{1}{2^m}}\ ;\ m=0,1,2,...,L, \qquad (4)$$

where, $T_{\frac{1}{2^m}}$ is defined in Definition 4.

Proof
Images of the size of $\mathbf{X}_{\frac{1}{2^m}}$ contain only contracted versions of range blocks limited to resolution m or greater. A block limited to resolution $k<m$ will become a single pixel in $\mathbf{X}_{\frac{1}{2^k}}$ and will disappear in the next contraction. We will prove that at any resolution, every range block in the fixed point **at this resolution** can be estimated with no error, using the transformation T. To do that, we will assume that an exact representation exists in $\mathbf{X}_{\frac{1}{2^{m-1}}}$, and show that a representation in $\mathbf{X}_{\frac{1}{2^m}}$ (for $1\leq m\leq L$) has also no errors. According to the theorem's conditions, for $m=1$ the representation $\mathbf{X}_{\frac{1}{2^{m-1}}}=\mathbf{X}_1$ is errorless. The case $m=0$ in equation (4) follows immediately from the last condition stated in the theorem. Hence we consider from now on only values of m which satisfy $m\geq1$.
Let $\mathbf{R}_i\in\mathbf{X}_{\frac{1}{2^{m-1}}}$, $(m\leq L)$, be a contracted version of a range block in $\mathbf{X}_1$. Denote its size by $2^k\times2^k$. Two cases may exist:

1. $\mathbf{R}_i$ corresponds to a range block limited to resolution greater than m-1.
2. $\mathbf{R}_i$ corresponds to a range block limited to resolution m-1.

Suppose the first case is true, then a local transformation $t_i=\{a_i,r_i,j_i\}$ exists in $T_{\frac{1}{2^{m-1}}}$ (and in $T_{\frac{1}{2^m}}$ also) so that

$$\mathbf{R}_i=a_i\cdot\left\{\mathbf{D}_{j_i}-\bar{\mathbf{D}}_{j_i}\right\}+\mathbf{r_i} \qquad (5)$$

and $\mathbf{D}_{j_i}$ exists in $\mathbf{X}_{\frac{1}{2^m}}$ (being a contraction of a domain block in $\mathbf{X}_{\frac{1}{2^{m-1}}}$). Applying the spatial contraction operator φ on both sides of the equation, yields:

$$\begin{aligned}\varphi\{\mathbf{R}_i\} &= \varphi\left\{a_i\left\{\mathbf{D}_{j_i}-\bar{\mathbf{D}}_{j_i}\right\}+\mathbf{r_i}\right\}\\ &= a_i\left\{\varphi\{\mathbf{D}_{j_i}\}-\varphi\{\bar{\mathbf{D}}_{j_i}\}\right\}+\varphi\{\mathbf{r_i}\}\\ &= a_i\left\{\varphi\{\mathbf{D}_{j_i}\}-\overline{\varphi\{\mathbf{D}_{j_i}\}}\right\}+\varphi\{\mathbf{r_i}\} \qquad (6)\end{aligned}$$

$\varphi\{\mathbf{R}_i\}$ is a contracted version of the same range block which relates to $\mathbf{R}_i$, and is a range block in $\mathbf{X}_{\frac{1}{2^m}}$. $\varphi\{\mathbf{D}_{j_i}\}$ is a contracted domain block that exists in $\mathbf{X}_{\frac{1}{2^{m+1}}}$ for $k>1$ by the terms of the theorem. $\varphi\{\mathbf{r_i}\}$ is a block of size $2^{k-1}\times2^{k-1}$ with constant value equals to r_i (which is given by the proper t_i). Since t_i (which is also in $T_{\frac{1}{2^m}}$) has all the needed parameters, the estimation of $\varphi\{\mathbf{R}_i\}$ in $\mathbf{X}_{\frac{1}{2^m}}$ from $\varphi\{\mathbf{D}_{j_i}\}$ is errorless. We draw the attention to the last equality of (6), *which holds only when the spatial contraction operator is commutative with the averaging operation.*
For $k=1$, a block $\varphi\{\mathbf{D}_{j_i}\}$ does not necessarily exist in $\mathbf{X}_{\frac{1}{2^{m+1}}}$, but in this case the block $\mathbf{D}_{j_i}$ is of the size 2×2,

therefore its contraction is 1×1 which is equal to its average, and equation (6) reduces to the form $\varphi\{\mathbf{R}_i\} = \varphi\{\mathbf{r}_i\}$, and since $\varphi\{\mathbf{R}_i\}$ is a 1×1 block, the estimation is, again, errorless.

Suppose now the second case to be true, i.e. $\mathbf{R}_i$ corresponds to a range block limited to resolution m-1. Hence its size in $\mathbf{X}_{\frac{1}{2^{m-1}}}$ must be 1×1, resulting in $\mathbf{R}_i = \mathbf{r}_i$ (where r_i is given by t_i which exists in all resolutions of the transformation T, until resolution m-1, including m-1).

Due to the nature of the quadtree algorithm it is guaranteed that if $\mathbf{R}_i$ is a 1×1 block, then it has three neighbors of the same size, so that a new block, $\mathbf{R}_{[0]}$ of size 2×2 may be defined, and $\mathbf{R}_{[0]}$ is composed of four 1×1 blocks $\mathbf{R}_{[1]}, \mathbf{R}_{[2]}, \mathbf{R}_{[3]}, \mathbf{R}_{[4]}$ ($\mathbf{R}_i$ is one of them).

$\mathbf{R}_{[0]}$ is a contracted version of a block limited to resolution m (though it is not a range block itself in $T_{\frac{1}{2^{m-1}}}$, but its contraction is a range block in $T_{\frac{1}{2^m}}$), and therefore $\varphi\{\mathbf{R}_{[0]}\}$ is a 1×1 range block that exists in $\mathbf{X}_{\frac{1}{2^m}}$. By the definition of the spatial contraction operator, we get:

$$\begin{aligned} \varphi\{\mathbf{R}_{[0]}\} &= \frac{\mathbf{R}_{[1]}^{1\times1} + \mathbf{R}_{[2]}^{1\times1} + \mathbf{R}_{[3]}^{1\times1} + \mathbf{R}_{[4]}^{1\times1}}{4} \\ &= \frac{\bar{\mathbf{R}}_{[1]} + \bar{\mathbf{R}}_{[2]} + \bar{\mathbf{R}}_{[3]} + \bar{\mathbf{R}}_{[4]}}{4} \\ &= \frac{r_{[1]} + r_{[2]} + r_{[3]} + r_{[4]}}{4} \end{aligned} \quad (7)$$

Since from Definition 4 the consequence is that $r_{[0]} = \frac{r_{[1]}+r_{[2]}+r_{[3]}+r_{[4]}}{4}$, then again, the estimation of $\varphi\{\mathbf{R}_{[0]}\}$ in $\mathbf{X}_{\frac{1}{2^m}}$ is without any error.

We conclude by the fact that the two kinds of $\varphi\{\mathbf{R}_i\}$ ($\mathbf{R}_i$ corresponding to range blocks limited to resolutions m-1 or greater) cover all of $\mathbf{X}_{\frac{1}{2^m}}$, hence in every level of resolution $m = 0, 1, ..., L$: $\mathbf{X}_f\left\{T_{\frac{1}{2^m}}\right\} = \mathbf{X}_{\frac{1}{2^m}}$. □

4 DEMONSTRATION

Using a fixed grid, i.e., fixed size range blocks, one will usually need to use range blocks of small size, like 4×4 or 8×8, to achieve good quality of the reconstructed image. Fixed grid compression results are demonstrated in Fig. 2. As mentioned above, quadtree adapts itself to the image, thus obtaining a lower rate while retaining quality. The reconstructed image is depicted in Fig. 3 on the right. A series of images obtained by IZI operations, resulting in the reconstructed image, is also depicted in the same figure. The quadtree map used for the encoding is depicted in Fig. 4.

As these figures show, the quadtree image is having a quality very close to the fixed grid image with small 4×4 range blocks, while at less than half the rate.

Using the quadtree partitioning of Fig. 4, a total of 1M additions and 333K multiplication operations are needed for the hierarchical decoder to obtain the reconstructed 512×512 image in Fig. 3. In comparison, using the same quadtree partitioning and assuming that six iterations are performed (as typically used) with the iterative decoder, then 11M additions and 1.6M multiplications will be needed for the reconstruction.

5 CONCLUSIONS

The generalized fast decoding algorithm was shown to reach the fixed point of a fractal transformation T_1 in a finite number of iterations, even if quadtree partitioning is used. We note that usually the iterative decoder doesn't reach the fixed point in a finite number of iterations. In section 4, the hierarchical decoder was shown to provide a reduction in computations by a factor of 11 in additions and a factor of about 5 in multiplications, as compared to an iterative decoder.

It was also seen that in order to deal with a transformation that uses blocks of different sizes, the transformation itself had to be changed when going to lower resolutions. The reason for this is that blocks that were used by T_1 no longer exist in low resolutions, due to the discrete grid of pixels, and T_1 cannot be applied to this type of image.

An implicit advantage that was not mentioned before and gives hierarchical decoding algorithms more power, is the abandoning of the contraction condition on the scale parameter (i.e. $|a_i| < 1$), since convergence is no longer an issue.

The partitioning method is irrelevant to the fractal encoder, but when a fast hierarchical decoder is desired, some partitioning methods cannot be used. A method which cannot be used is, for example, HV partitioning (Horizontal and Vertical tiling[4]). However, when the image sides are 2 to the power of an integer, and rectangular range blocks of similar shapes are sliced in half only in one coordinate, fast decoding is feasible. This will be expanded upon in an extended publication.

References

[1] G E ϕien, S Lepsϕy, and T A Ramstad, 1991, "An inner product space approach to image coding by contractive transformations", in ICASSP, 2773-2776.

[2] Z Baharav D Malah and E Karnin, July 1993, "Hierarchical interpretation of fractal image coding and its application to fast decoding", in Digital Signal Processing Conf., Cyprus.

[3] Z Baharav, D Malah and E Karnin, 1995, "Hierarchical Interpretation of Fractal Image Coding and Its Applications", Ch.5 in [4] below.

[4] Fisher Y, ed., 1995, "Fractal Image Compression - Theory and Application", Springer-Verlag, NY.

[5] A Jacquin, Oct. 1993, "Fractal Image Coding: A Review", Proc. of the IEEE, 81, 1451-1465.

[6] A Jacquin, Jan. 1992, "Image Coding Based on Fractal Theory of Iterated Contractive Image Transformation", IEEE Trans. on Image Processing, 1, 18-30.

Figure 2: **On the left, the original 512×512 Lena picture. In the middle the reconstructed image with 4×4 range blocks (at 1.625bpp. PSNR=36.6dB). On the right, reconstructed image with 8×8 range blocks (at 0.375bpp. PSNR=30.2dB.)**

Figure 3: **The four images on the left are the transformation fixed points at different resolutions ($X_{\frac{1}{2^m}}, m = 1, ..., 4$). These images are obtained by IZI operations one from the other, where every pixel in the leftmost image is the average of a 16×16 block (with no overlap) in the original image (and also in the fixed point). The last step of the algorithm yields the image on the right, which is the reconstructed image (at 0.772bpp. PSNR=35.4dB). All images demonstrate steps in the the fast decoding algorithm described in section 3.**

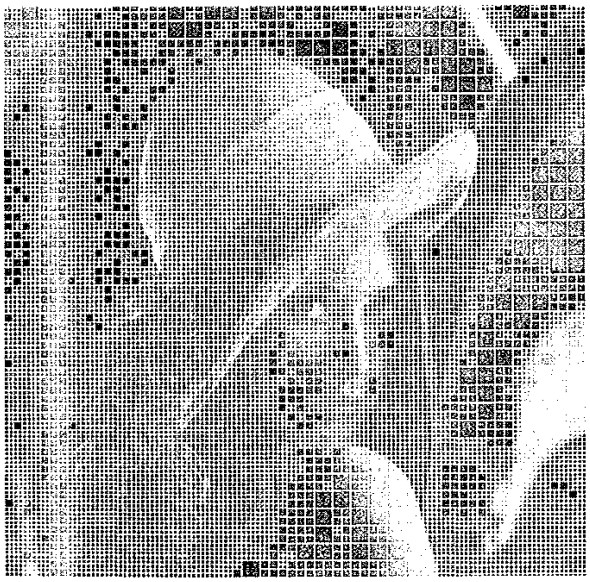

Figure 4: **The corresponding quadtree map used to encode the image. Smallest blocks are 4×4 and biggest are 16×16.**