Image Compression via Improved Quadtree Decomposition Algorithms

Eli Shusterman and Meir Feder, *Senior Member, IEEE*

***Abstract*—Quadtree decomposition is a simple technique used to obtain an image representation at different resolution levels. This representation can be useful for a variety of image processing and image compression algorithms. This paper presents a simple way to get better compression performances (in MSE sense) via quadtree decomposition, by using:**

- **Near to optimal choice of the threshold for quadtree decomposition.**
- **Bit allocation procedure based on the equations derived from rate-distortion theory.**

The rate-distortion performance of the improved algorithm is calculated for some Gaussian field, and it is examined vie simulation over benchmark gray-level images. In both these cases, significant improvement in the compression performances is shown.

I. Introduction

QUADTREE (QT) decomposition is a simple technique for image representation at different resolution levels. This representation is successfully used in binary image compression algorithms. Recently, QT decomposition has been used as a part of image sequence compression algorithms, [17]–[19]. QT decomposition for coding of gray-level images is attractive for a number of reasons:

- Relative simplicity compared to other methods (e.g., DCT-based coding), which makes it an attractive method for applications such as video and HDTV compressions.
- The *adaptivity* of the decomposition. The decomposition divides the image into regions with size depending on the activity in the region. The compression performance is thus adapted to the various image regions.
- The useful output of the decomposition. The decomposition actually results in a kind of image segmentation. This segmentation can be used for a variety of different image processing applications, e.g., pattern recognition, [11].

However, so far, the rate-distortion (R-D) performance of QT based compression algorithms for gray-level images has been poorer than other popular compression techniques such as transform-based compressions (e.g., DCT).

Little attention has been paid to the study of the threshold and the bit allocation influence on the decomposition performance. This paper tries to fulfill the gap. The results of the present study suggest two modifications in the *original QT* decomposition coding algorithm. The suggested modifications were analyzed for an autoregressive Gaussian field and tested by simulations on real gray-level images. In both cases these modifications significantly improved the R-D performance of the algorithm with a minor increase in its complexity. The simulation results show that a proposed algorithm performs better than transform coding or subband coding [22], both followed by scalar quantizer. For some images the algorithm performance is competitive even with the two mentioned coding techniques, followed by vector quantizer.

The paper is organized as follows: In Section II the standard QT decomposition algorithm is described. Section III introduces modifications of the standard QT algorithm, which improve its performance. This is the main contribution of the paper. In Section IV the R-D performance of the improved QT decomposition coding algorithm is calculated for an autoregressive Gaussian field. Section V presents simulation results for real gray-level images and Section VI concludes this work.

II. QT Decomposition

A natural gray-level image usually can be divided into different size regions with a variable amount of details and information. Such segmentation of the image is useful for efficient coding of image data. QT decomposition is a powerful technique which divides the image into 2-D *homogeneous* (in the property of interest) regions, i.e., produces the segmentation.

The decomposition builds a tree. Each tree node has four children and it is associated with a uniquely defined region of the image. It is obvious that the root is associated with the whole image. QT decomposition can be done either by top-down or bottom-up procedures. In Fig. 1, both top-down and bottom-up QT decomposition procedures are illustrated. It is well known, and also demonstrated by this simple example, that the bottom-up procedure is superior; therefore it is preferred for usage in the suggested algorithm.

When QT decomposition is used for image compression, the resulting tree is coded. The coding procedure includes coding of the tree structure information and coding of the leaf information. Let assign "1" to the parent node and "0" to the leaf. To each leaf a parameter (or parameters) that describes the intensity of the corresponding subimage will also be assigned. Obviously, the image pixels are always leaves, so the tree structure coding can be stopped one level before the bottom level. Fig. 2 demonstrates the tree and the resulting code of the example in Fig. 1.

Manuscript received November 3, 1992; revised September 15, 1993. The associate editor coordinating the review of this paper and approving it for publication was Prof. Michel Barlaud.

The authors are with the Department of Electrical Engineering-Systems, Faculty of Engineering, Tel-Aviv University, Ramat-Aviv, Tel-Aviv 69978, Israel.

IEEE Log Number 9215225.

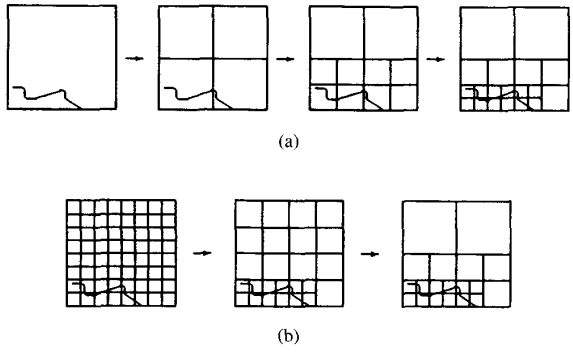
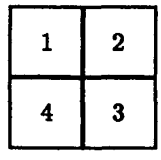
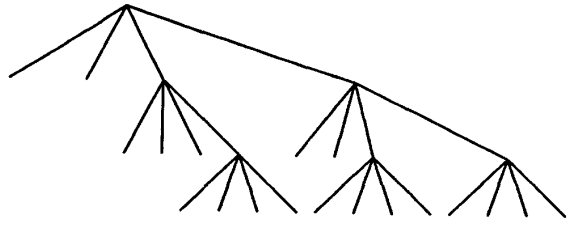


Fig. 1. Quadtree decomposition procedures. (a) Top-down procedure. (b) Bottom-up procedure.



(a)



(b)

Fig. 2. The quadtree and the corresponding code. (a) Subimage coding order. (b) The resulting tree; tree code: QTC = 1 − 0011 − 0001 0011.

So far, the general conventions have been briefly explained. Now more specific parts of the QT algorithm will be described. Suppose the image size is $2^n \times 2^n$; it can then be represented at $n+1$ levels of resolution. Every pixel at every resolution level has its own intensity level. The parent node intensity is a mean value of its children nodes intensities, and the test examines the error of this representation in the property of interest. At each level, except the bottom, the node intensity is calculated according to

$$\begin{aligned} &\text{for } i = 1, \cdots, n; \\ &\text{for } k, l = 0, \cdots, 2^{n-i} - 1; \\ &x_i(k,l) = \frac{1}{4} \sum_{j=0}^{1} \sum_{m=0}^{1} x_{i-1}(2k+j, 2l+m) \end{aligned} \tag{1}$$

where x_i denotes the pixel intensity at resolution level i. Now the test must be applied.

Several homogeneity tests have been introduced in the literature [20] and more tests can be defined, depending on the property of interest in image representation. However, in most applications the simple *absolute difference* test is used. Therefore only this test will be referenced. Other tests can be analyzed in the same way. Let T be the threshold. The test is positive if

$$\bigcap_{j,m=0}^{1} \mid x_i(k,l) - x_{i-1}(2k+j, 2l+m) \mid \leq T = \text{ True.} \tag{2}$$

The standard QT decomposition algorithm can be summarized as follows.

Step 1: Let $i = 1; N = 2^{n-1}$;

Step 2: For $k, l = 0, \cdots, N-1$;
IF for $j, m = 0, 1$ all $x_{i-1}(2k+j, 2l+m)$ are leaves
calculate $x_i(k,l)$ according to (1)
perform the test according to (2)
IF the test is $TRUE$ $x_i(k,l)$ is a leaf,
ELSE $x_i(k,l)$ is a node.
next k, l.

Step 3: IF no leaves were produced by Step 2 STOP,
ELSE $N = \frac{N}{2}; i = i+1$; go to Step 2.

The described algorithm has been used in many image compression applications, e.g., [14]–[21]; however, as it will be shown in the next section, by a few modifications the performance of the algorithm for image compression can be significantly improved.

III. The Improved Compression Algorithm Based on QT Decomposition

A. General

In contrast to most image compression algorithms, the described QT algorithm does not need much computation power. However, as mentioned before, the so far poor rate distortion performance of the algorithm (for gray-level images) discards all its advantages. In addition to the lack of performance, the algorithm creates *blocking* in the reconstructed image. To make the algorithm useful, both these problems must be solved. The main result of this paper is that R-D performance can be improved by *optimal* threshold adjustment in the homogeneity test and by *optimal* bit allocation for leaves coding.

For further discussion, the following variables are defined: N_i is a number of pixels at level i that did not propagate to level $i+1$. Define the *empirical probability* of finding a pixel at level i as

$$p_i = \frac{N_i}{4^{n-i}}. \tag{3}$$

The pixel propagation probability to resolution level 0 is defined as $q_0 = 1$. The *empirical propagation probability* from level $i-1$ to level i is defined by a recursive equation

$$q_i = \begin{cases} 1 & i = 0; \\ q_{i-1} - p_{i-1} & i = 1, \cdots, n. \end{cases} \tag{4}$$

Note that $q_n = q_{n-1} - p_{n-1} = p_n$. The compressed tree contains L_{qt} leaves, where, by definitions above

$$L_{qt} = \sum_{i=0}^{n} 4^{n-i} p_i. \tag{5}$$

The uncompressed tree contains $\frac{4^n-1}{3}$ nodes, so the tree structure code of the complete tree consists of $\frac{4^n-1}{3}$ *ones*. During the compression, the propagation from level 0 to level 1 converts a number of nodes to leaves. The code remains $\frac{4^n-1}{3}$ bits but includes *zeros* in it. During further compression, the pixel propagation to resolution level i reduces the code length by $4^{n-i+1}q_i$ bits. Therefore the compressed tree contains N_{qt} nodes and leaves that are sufficient tree structure information, i.e., N_{qt} bits must be assigned for the tree structure code, where

$$N_{qt} = \frac{4^n - 1}{3} - \sum_{i=1}^{n} 4^{n-i+1} q_i. \tag{6}$$

The rate R_{qt} of the tree structure code, assuming the straight forward coding, is given by

$$R_{qt} = \frac{N_{qt}}{4^n}. \tag{7}$$

In the example of Fig. 2, $n = 3$ and

$$\begin{aligned} p_0 &= \tfrac{12}{64} = \tfrac{3}{16}; & q_0 &= 1; \\ p_1 &= \tfrac{5}{16}; & q_1 &= 1 - p_0 = \frac{13}{16}; \\ p_2 &= \tfrac{2}{4} = \tfrac{1}{2}: & q_2 &= q_1 - p_1 = \frac{1}{2}; \\ p_3 &= 0; & q_3 &= q_2 - p_2 = 0 \end{aligned}$$

$$L_{qt} = 64 \times \frac{3}{16} + 16 \times \frac{5}{16} + 4 \times \frac{1}{2} = 19;$$
$$N_{qt} = \frac{64-1}{3} - 16 \times \frac{1}{2} = 21 - 8 = 13;$$

$$R_{qt} = \frac{13}{64} \simeq .2[\text{bits/pixel}].$$

Although this rate may be reduced further by a proper coding, this coding is not really required, since, as evident from the example, this rate is small (would be about 10% – 20% of the total rate).

B. How to Choose the Threshold

To improve the algorithm performance it is first suggested to use different threshold values at each resolution level. The procedure to determine the threshold values will be described below. Let T_i denote the threshold value at level i.

Lemma 1: The mean-square error (MSE) of the subimage representation by a leaf at QT level i is upper bounded by:

$$\text{MSE}_i \le \sum_{j=1}^{i} T_j^2. \tag{8}$$

The proof of Lemma 1 is given in the appendix. As a universal bound it is sharp, since, as shown in Fig. 3, a subimage that achieves the bound can be synthesized.

The whole image is represented by a number of leaves at different levels, where each leaf is a representation of the correspondent subimage. Therefore, the total MSE of the reconstructed image is bounded by

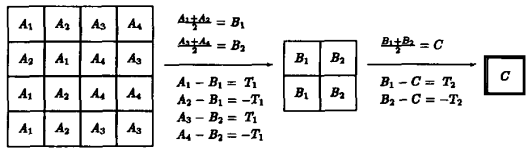


Fig. 3. The synthetic subimage that achieves the MSE bound given in Lemma 1. A_i, B_i, and C are pixel values. T_i is the threshold at level i.

$$\text{MSE}_{qt} = \sum_{i=1}^{n} p_i \text{MSE}_i \le \sum_{i=1}^{n} p_i \sum_{j=1}^{i} T_j^2. \tag{9}$$

Expressing p_i by q_i in (4) and substituting p_i in (9) gives

$$\begin{aligned} \text{MSE}_{qt} &= \sum_{i=1}^{n} (q_i - q_{i+1})\, \text{MSE}_i \le \sum_{i=1}^{n} (q_i - q_{i+1}) \sum_{j=1}^{i} T_j^2 \\ &= q_1 T_1^2 + q_2 T_1^2 + q_2 T_2^2 + \cdots + q_n T_1^2 + \cdots + q_n T_n^2 \\ &\quad - q_2 T_1^2 - q_3 T_1^2 - q_3 T_2^2 - \cdots - q_n T_1^2 - \cdots - q_n T_{n-1}^2 \\ &\quad - q_{n+1} \sum_{j=1}^{n} T_j^2 = \sum_{i=1}^{n} q_i T_i^2 - q_{n+1} \sum_{j=1}^{n} T_j^2. \end{aligned} \tag{10}$$

Level $n+1$ does not exist, so $q_{n+1} = 0$. Therefore, the total MSE is bounded by

$$\text{MSE}_{qt} \le \sum_{i=1}^{n} q_i T_i^2. \tag{11}$$

The goal is to choose threshold values $\{T_i\}$ that minimize the MSE under a constraint of a fixed number of leaves. However, the relation between the threshold values and the MSE is complicated and cannot be expressed in a closed form. Thus, we suggest to choose threshold values that minimize the MSE bound (11) while the number of leaves is constant or equivalently to minimize the number of leaves while the MSE bound is constant. By substituting (4) in (5) and writing the summation explicity, the expression for L_{qt} becomes

$$\begin{aligned} L_{qt} &= 4^n(1 - q_1) + \sum_{i=1}^{n} 4^{n-i}(q_i - q_{i+1}) \\ &= 4^n\Big(1 - q_1 + \frac{q_1}{4} - \frac{q_2}{4} + \frac{q_2}{16} - \cdots \\ &\quad \cdots + \frac{q_{n-1}}{4^{n-1}} - \frac{q_n}{4^{n-1}} + \frac{q_n}{4^n}\Big) \\ &= 4^n\left[1 - \frac{3}{4}\Big(q_1 + \frac{q_2}{4} + \cdots + \frac{q_n}{4^{n-1}}\Big)\right] \\ &= 4^n\left(1 - 3\sum_{i=1}^{n} \frac{q_i}{4^i}\right). \end{aligned} \tag{12}$$

Then

$$q_1 = \frac{4}{3}(1 - 4^{-n} L_{qt}) - \sum_{i=2}^{n} 4^{1-i} q_i. \tag{13}$$

The expression for q_1 substituted in (11) gives

$$\mathrm{MSE}_{qt} \le \frac{4}{3}(1 - 4^{-n}L_{qt})T_1^2 + \sum_{i=2}^{n} q_i(T_i^2 - 4^{1-i}T_1^2). \quad (14)$$

The first term of (14) is always positive and if L_{qt} is given, the term depends only on T_1. Suppose that T_1 is a given parameter. Then, if the second term is minimized, the MSE bound is also minimized. The second term is not positive if

$$2 \le i \le n \qquad T_i \le 2^{1-i}T_1. \quad (15)$$

Obviously, one can claim that there exists some set of numbers $\hat{\Upsilon} \in \{\varepsilon_i, \varepsilon_i \in [0,1]\}_{i=2}^{n}$ such that

$$2 \le i \le n \qquad T_i = 2^{1-i}\varepsilon_i T_1$$
$$\sum_{i=2}^{n} 4^{1-i}q_i T_1^2(\hat{\varepsilon}_i^2 - 1) = \min_{\text{all}\Upsilon}\left\{\sum_{i=2}^{n} 4^{1-i}q_i T_1^2(\varepsilon_i^2 - 1)\right\}. \quad (16)$$

The probability q_i depends on T_i and on the image statistics. Thus, each image gives different $\hat{\Upsilon}$. Furthermore, to find such $\hat{\Upsilon}$ for every image can be very expensive in calculations. Therefore, a suboptimal but universal solution (e.g., solution that holds for any image no matter what its statistic is) is given by

$$2 \le i \le n \qquad T_i = 2^{1-i}T_1. \quad (17)$$

This is the procedure we suggest for threshold selection. It depends on an arbitrary parameter T_1 that can control the R-D tradeoff. At each level the threshold is a factor of 2 smaller than its value at the previous level. A simple "intuitive" explanation can be given to this result. The contribution of the image area representation by a leaf at level i to total distortion is proportional to the size of the area (4^i). The best MSE is achieved if the distortion is distributed uniformly between all leaves. Thus, at the higher levels, lower distortion is allowable.

C. Bit Allocation

So far only distortion due to QT decomposition has been referenced. The additional distortion is created by quantizing the pixel value of the various leaves; the aim of this section is to show how to minimize this distortion by optimal bit allocation. R-D theory provides good solutions to the problem of optimal bit allocation for an independent vector source [6]. Each resolution level can be handled as an independent source, but in fact all levels, except the bottom, are some combination of the previous level, so better performance can be achieved if the dependencies are taken into account.

The rate distortion function of a source (a random variable), denoted $R(D)$, is bounded [1]–[3] by

$$R_L(D) = \frac{1}{2}\log\frac{P}{D} \le R(D) \le \frac{1}{2}\log\frac{\sigma^2}{D} \quad (18)$$

where $\frac{1}{2}\log\frac{\sigma^2}{D}$ is the R-D function of the Gaussian source with the same varience σ^2 as the given source and $R_L(D)$ is the R-D function of the Gaussian source with the same entropy. P in (18) denotes the entropy power of the source. Suppose each level is an independent source; then according to (18) the R-D function of the every source is bounded by

$$\frac{1}{2}\log\frac{P_i}{D_i} \le R_i(D_i) \le \frac{1}{2}\log\frac{\sigma_i^2}{D_i} \quad (19)$$

for the total leaves rate

$$\sum_{i=0}^{n} 4^{-i}p_i\frac{1}{2}\log\frac{P_i(x)}{D_i} \le R(D)$$
$$= \sum_{i=0}^{n} p_i\frac{R_i(D_i)}{4^i} \le \sum_{i=0}^{n} 4^{-i}p_i\frac{1}{2}\log\frac{\sigma_i^2}{D_i} \quad (20)$$

where we recall that $P_i(x)$ is the entropy power of the random variable x corresponding to the intensity at level i and σ_i^2 is its variance. The distortion due to quantization is

$$D = \sum_{i=0}^{n} p_i D_i. \quad (21)$$

Now, let us find a set of D_i's that minimizes the upper bound in (20), while D is constant and (21) is a constraint. The differentiation of the upper bound in (20) with a Lagrange multiplier gives

$$D_i = \frac{4^{n-i}D}{L_{qt}}. \quad (22)$$

The minimization of the lower bound in (20) with the same constraint (21) gives the same solution (22). Both, the upper and the lower bounds of the R-D function reach their minimum at the same point. Thus, it is a good reason to believe that the function itself reaches the minimum at or near this point.

Finally, the near to optimal bit allocation is

$$B_i = \frac{1}{2}\log\frac{\sigma_i^2 L_{qt}}{4^{n-i}D} \quad (23)$$

and the leaves rate is

$$R_B = \sum_{i=0}^{n} 4^{-i}p_i B_i. \quad (24)$$

D. Summary: The Improved Algorithm for Image Compression

We now summarize the proposed modifications, and introduce the improved algorithm. The homogeneity test of (2) is modified to

$$\bigcap_{j,m=0}^{i} |\, x_i(k,l) - x_{i-1}(2k+j, 2l+m)\,| \le T_i = \text{True}. \quad (25)$$

The new image compression algorithm via QT decomposition is as follows

Step 1: Choose T_1. Let $i = 1; N = 2^{n-1}$;

Step 2: For $k, l = 0, \cdots, N-1$:

IF for $j, m = 0, 1$ all $x_{i-1}(2k+j, 2l+m)$ are leaves

calculate $x_i(k,l)$ according to (1)

perform the test according to (25)

IF the test is $TRUE$ $x_i(k,l)$ is a leaf,

ELSE $x_i(k,l)$ is a node.

next k, l.

Step 3: IF no leaves were produced by Step 2 goto Step 4, ELSE $N = \frac{N}{2}; i = i + 1; T_i = \frac{T_{i-1}}{2}$; goto Step 2.
Step 4: Code the tree structure information as it was described in Section 2.
Step 5: Calculate L_{qt}; Choose a desired distortion level D for leaves quantization; $i = 0$;
Step 6: Calculate leaves mean m_i and variance σ_i^2.
Step 7: Allocate bits for level i leaves according to (23).
Step 8: Quantize level i leaves.
Step 9: $i = i + 1$; IF $i > n$ STOP, ELSE goto Step 6.

In this work the Lloyd-Max quantizer [4], [5], designed for a Gaussian distribution, was chosen for the quantization in Step 8. The overhead information includes means and variances of all levels. The total rate is given by

$$R_T(D_{qt} + D) \simeq R_{qt} + R_B(D) \tag{26}$$

and the total distortion is upper bounded

$$D_T(R_T) \leq \text{MSE}_{qt} + D. \tag{27}$$

IV. Example: An Autoregressive Gaussian Field

In this section we assume a specific source statistics, and compare the performance of the proposed improved algorithm to the original QT compression algorithm. If the image statistic is known, in several cases the R-D performance of the original algorithm can be calculated explicitly as described in [21]. We follow the technique in [21] to get the performance of the improved algorithm, as shown bellow.

Specifically, we calculate the R-D performance for the case where the image is supposed to be a 2-D autoregressive Gaussian field with zero mean and the following autocorrelation function

$$R(k,l) = \sigma^2 \exp\left(-\frac{|k| + |l|}{d}\right) \tag{28}$$

where d is the correlation distance. The correlation factor between two adjacent pixels is given by

$$\rho = \frac{R(k+1,l)}{R(k,l)} = \frac{R(k,l+1)}{R(k,l)} = \exp\left(-\frac{1}{d}\right). \tag{29}$$

The autocorrelation function can be expressed in the following matrix form

$$R = \sigma^2 \begin{pmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{2^n-1} \\ \rho & \rho^2 & \rho^3 & \cdots & \rho^{2^n} \\ \rho^2 & \rho^3 & \rho^4 & \cdots & \rho^{2^n+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho^{2^n-1} & \cdots & \cdots & \cdots & \rho^{2(2^n-1)} \end{pmatrix}. \tag{30}$$

Without loss of generality the variance σ^2 is chosen equal to unity. The correlation factor ρ varies for real gray-level images in the range of 0.9–0.99. The R-D performance is calculated for:

1) *Original* quadtree decomposition algorithm.
2) Quadtree decomposition with the logarithmic threshold of (17).

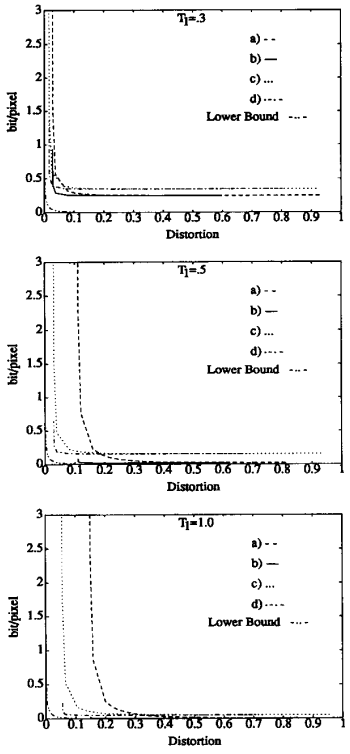


Fig. 4. R-D functions for the autoregressive Gaussian field. $\rho = .98$

3) *Original* quadtree decomposition algorithm. The optimal bit allocation of (23) is performed.
4) Quadtree decomposition with the logarithmic threshold of (17). The optimal bit allocation of (23) is performed.

The threshold T_1 and the correlation factor ρ are given parameters, the resulting R-D functions are shown in Figs. 4–6. The examination of these graphs leads to a number of conclusions. First, the proposed bit allocation significantly improves the R-D performance of the compression for either the constant or the logarithmic threshold assignment. Second, the R-D function of the algorithm with the logarithmic threshold assignment intersects with the R-D function of the original algorithm. Before the intersection, the first algorithm performs better; after the intersection, the second. However the intersection occurs above the distortion range that is allowable for a good image compression algorithm. Therefore the proposed threshold assignment improves the R-D performance in the interesting region. It should be noted that if the threshold T_1 is low, then most of the pixels do not propagate to levels higher than 1. Thus, if T_1 is low, the performance does not depend on how the threshold for the next level is chosen.

V. Simulation Results for Test (Benchmark) Images

A. Distortion Measure

Before presenting the simulation results let us define the distortion measure. Different measures have been used by

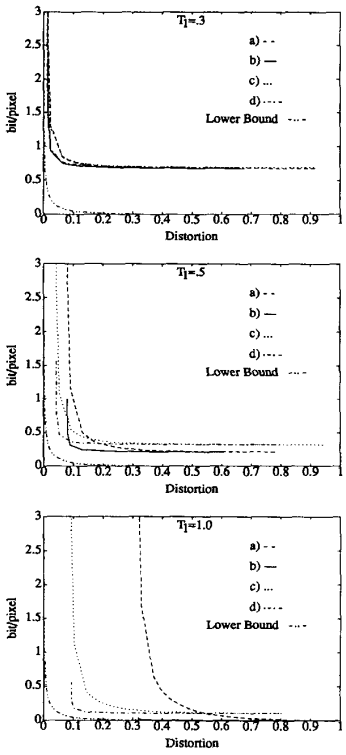


Fig. 5. R-D functions for the autoregressive Gaussian field. $\rho = .95$

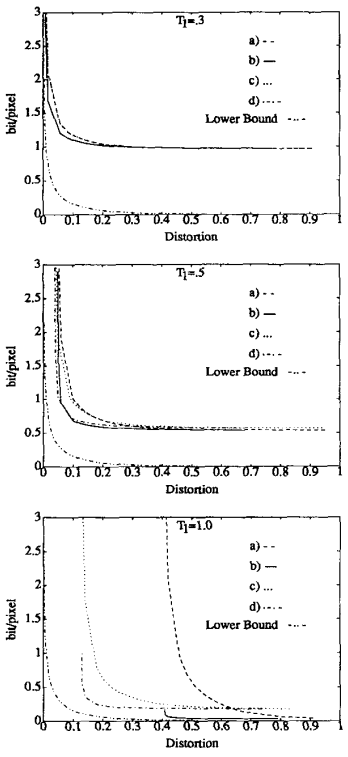


Fig. 6. R-D functions for the autoregressive Gaussian field. $\rho = .90$

different researchers. For comparison purpose to other reported work, we use the measure peak signal-to-noise ratio (PSNR) defined as follows: let $x(i,j)$ be a pixel at the i,j coordinates of the source image and let $y(i,j)$ be a pixel at the i,j coordinates of the reconstructed image. The image size is $2^n \times 2^n$ pixels and 8 bit/pixel.

$$\text{PSNR} = 10 \log \left\{ \sum_{i=0}^{2^n-1} \sum_{j=0}^{2^n-1} \frac{256^2}{[x(i,j) - y(i,j)]^2} \right\}. \quad (31)$$

B. Reconstruction Filter

So far in this work, only the compression procedure was of interest. However, an important part of the algorithm is the reconstruction procedure, which may influence the reconstructed image quality that is not measured directly by SNR value. The goal of the reconstruction procedure is to expand each tree leaf at some level i to a number of leaves at level 0. The reconstruction filter is chosen to be of the following form

$$F = \begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix} \quad (32)$$

and the reconstruction of a pixel at resolution level $i-1$ is done according to

$$\text{for} \quad m,n = 0,1$$
$$x_{i-1}(2k+m, 2l+n) = R_m F R_n X_i(k,l) \quad (33)$$

where

$$X_i(k,l) = \begin{pmatrix} x_i(k-1,l-1) & x_i(k-1,l) & x_i(k-1,l+1) \\ x_i(k,l-1) & x_i(k,l) & x_i(k,l+1) \\ x_i(k+1,l-1) & x_i(k+1,l) & x_i(k+1,l+1) \end{pmatrix}$$
$$R_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \; R_1 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}. \quad (34)$$

The reconstruction filter is designed to reduce the *blocking* in the reconstructed image. The best way to test if the blocking exists is by a visual test. In the results reported below, the filter coefficients are taken from [19].

C. Results

The proposed compression algorithm is tested on two real gray-level images. The size of the source pictures is 256×256 and they are are shown in Fig. 8.

In the graph of Fig. 7, the R-D functions for "Lena" of four different quadtree based compression algorithms are plotted (the reported rate includes the overhead information, which is about 10% of the given value). The R-D function of the proposed algorithm is more than 5 db above the R-D function of the original algorithm. Table I compares PSNR values of two quadtree coders for test images at several rates.

These results make obvious the fact that the proposed changes lead to significant performance improvement in the compression algorithm based on quadtree decomposition.

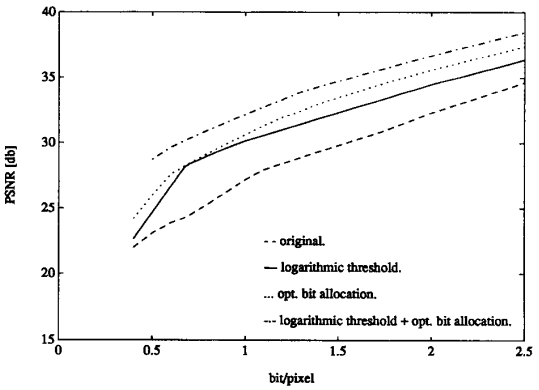


Fig. 7. R-D function for "Lena."

(a) (b)

Fig. 8. Test pictures. (a) "Lena." (b) "Tree."

Next, it is interesting to ask: whether the proposed improved algorithm is competitive with other compression techniques, e.g., DCT-based techniques. To answer this question the algorithm performance is compared with several popular techniques that are described in [22]. The comparison results are shown in Tables II and III, and indicate that the proposed algorithm performance is much better than popular transform coding with scalar quantization and is competitive with subband coding with scalar quantization within the bands. Furthermore, for some images (like "Lena") the proposed algorithm is competitive with transform or subband coding with vector quantization. Finally, the resulting images at different compression rates are shown in Figs. 9 and 10.

TABLE I
COMPARISON OF TWO QUADTREE CODERS AT SEVERAL RATES

	"Lena"		"Tree"	
Rate	Old QT Alg.	New QT Alg.	Old QT Alg.	New Qt Alg.
0.5	23.08	28.91	20.34	25.34
0.67	24.21	30.36	21.75	26.95
1.0	27.18	32.55	24.42	29.36
2.0	32.29	37.19	29.57	33.87

TABLE II
PSNR[1] VALUES FOR SEVERAL CODERS AT rate = 0.5 bit/pixel

		Transform				Subband	
Image	QT	SQ	PVQ	AWPVQ	FSVQ	SQ	PVQ
"Lena"	28.91	24.87	27.62	28.41	29.91	26.55	28.37
"Tree"	25.38	23.37	26.05	26.67	27.99	26.27	27.13

[1]All PSNR values for transform and subband coders are according to M. E. Blain and T. R. Fischer [22]. Abbreviations are: QT: quadtree coder; SQ: scalar quantizer; PVQ: pyramid vector quantizer; AWPVQ: adaptive weighted PVQ; FSVQ: full-search vector quantizer; DPCM: differential pulse code modulation; ADPCM: adaptive DPCM.

TABLE III
PSNR[1] VALUES FOR "Lena" IMAGE AT DIFFERENT RATES

		Transform			Subband			
Rate	QT	SQ	PVQ	AWPVQ	SQ	PVQ	DPCM	ADPCM
0.5	28.91	24.87	27.62	28.41	26.55	28.37	—	—
0.67	30.36	25.97	28.66	29.93	27.12	29.48	29.4	30.9
1.0	32.55	27.07	31.67	32.2	28.15	31.37	31.4	32.5
2.0	37.19	30.67	36.66	—	31.2	36.65	35.4	36.6

[1]All PSNR values for transform and subband coders are according to M. E. Blain and T. R. Fischer [22]. Abbreviations are: QT: quadtree coder; SQ: scalar quantizer; PVQ: pyramid vector quantizer; AWPVQ: adaptive weighted PVQ; FSVQ: full-search vector quantizer; DPCM: differential pulse code modulation; ADPCM: adaptive DPCM.

VI. CONCLUSION

A practical and efficient image compression algorithm is proposed in this paper. This algorithm is based on quadtree decomposition and has all its advantages. On the other hand, it does not suffer from the poor R-D performance of standard QT-based algorithms and its performance is even superior to DCT-based compression algorithms with scalar quantization and competitive with DCT-algorithms with vector quantization. Thus the proposed algorithm is attractive for a variety of applications. The fact that the compression output is useful for image segmentation, edge enhancement and pattern recognition adds to the attractiveness of the algorithm. It is worth mentioning that other improvements of the quadtree decomposition have been suggested [13], [14]. We note, however, that our modifications can be incorporated with these algorithms as well. For example, our thresholding and bit allocation techniques can be applied to the "dual-domain" coders described in [14] and further improve their performance. Finally, the quadtree decomposition has close relations to Pyramid and Wavelet decomposition. The proposed modifications can be used in image compression algorithms based on these decompositions as well.

APPENDIX

Proof of Lemma 1: The proof is by induction. Let us start from level 1. According to the test (2) MSE_1 can be represented as

$$\mathrm{MSE}_1 = \frac{1}{4}T_1^2(\alpha_1^2(1) + \alpha_1^2(1) + \alpha_1^2(3) + \alpha_1^2(4)) \qquad (35)$$

where $\mid \alpha_1(i) \mid \leq 1$ and $\sum_{i=1}^{4} \alpha_1(i) = 0$. In the worst case $\mid \alpha_1(i) \mid = 1$, then $\mathrm{MSE}_1 = T_1^2$. Now, let us proceed to the

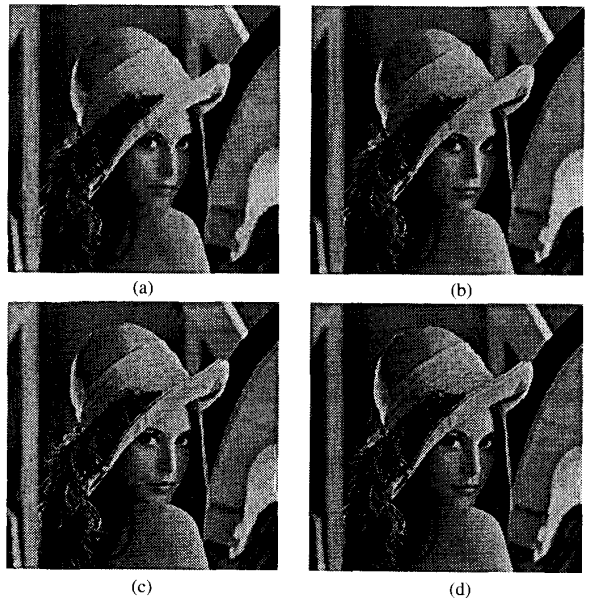


Fig. 9. Image "Lena" at several rates. (a) 0.5 bit/pixel, 28.91 dB. (b) 0.67 bit/pixel, 30.36 dB. (c) 1.0 bit/pixel, 32.55 dB. (d) 2.0 bit/pixel, 37.19 dB.

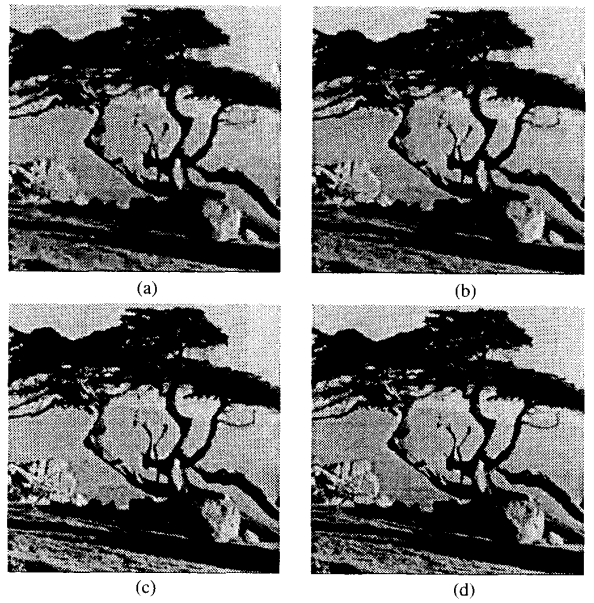


Fig. 10. Image "Tree" at several rates. (a) 0.5 bit/pixel, 25.34 dB. (b) 0.67 bit/pixel, 26.95 dB. (c) 1.0 bit/pixel, 29.36 dB. (d) 2.0 bit/pixel, 33.8 dB.

next level

$$\begin{aligned}\mathrm{MSE}_2 = \frac{1}{16}\sum_{i=1}^{4}&(\alpha_2(i)T_2 + \alpha_1(i,1)T_1)^2 \\ &+ (\alpha_2(i)T_2 + \alpha_1(i,2)T_1)^2 \\ &+ (\alpha_2(i)T_2 + \alpha_1(i,3)T_1)^2 \\ &+ (\alpha_2(i)T_2 + \alpha_1(i,4)T_1)^2\end{aligned} \tag{36}$$

where $|\ \alpha_2(i) \le 1, |\ \alpha_1(i,j)\ | \le 1$ and $\sum_{i=1}^{4}\alpha_2(i) = \sum_{j=1}^{4}\alpha_1(i,j) = 0$. After a few simple operations. Equation (36) becomes

$$\mathrm{MSE}_2 = \frac{1}{16}(4T_2^2)\sum_{i=1}^{4}\alpha_2^2(i) + T_1^2\sum_{i=1}^{4}\sum_{j=1}^{4}\alpha_1^2(i,j). \tag{37}$$

Again, in the worst case $|\ \alpha_2(i)\ | = |\ \alpha_1(i,j)\ | = 1$, then $\mathrm{MSE}_2 = T_1^2 + T_2^2$. Suppose that the relation $\mathrm{MSE}_i \le \sum_{j=1}^{i} T_j^2$ holds for the first $k-1$ levels. Now, let us prove that the relation holds for level k. The equation for the MSE at level k is

$$\begin{aligned}\mathrm{MSE}_k = \frac{1}{4^k}\sum_{i=1}^{4}&(\alpha_k(i)T_k + E_{k-1}(i,1)^2 \\ &+ (\alpha_k(i)T_k + E_{k-1}(i,2))^2 \\ &+ \cdots + (\alpha_k(i)T_k + E_{k-1}(i,4^{k-1}-1))^2 \\ &+ (\alpha_k(i)T_k + E_{k-1}(i,4^{k-1}))^2\end{aligned} \tag{38}$$

where $|\ \alpha_k(i)\ | \le 1$ and $\sum_{i=1}^{4}\alpha_k(i) = 0$. $E_{k-1}(i,j)$ denotes the difference between the pixel at level $k-1$ and the corresponding pixel at level 0. It is easy to show that: $\sum_{j=1}^{4^{k-1}} E_{k-1}(i,j) = 0$ and $\sum_{j=1}^{4^{k-1}} E_{k-1}^2(i,j) = \mathrm{MSE}_{k-1}(i)$. Equation (38) can be rewritten as

$$\begin{aligned}\mathrm{MSE}_k &= \frac{1}{4^k}\sum_{i=1}^{4}[4^{k-1}\alpha_k^2(i)T_k^2 + 2\alpha_k(i)\sum_{j=1}^{4^{k-1}}E_{k-1}(i,j) \\ &\quad + \sum_{j=1}^{4^{k-1}}E_{k-1}^2(i,j)] \\ &= \frac{1}{4}\sum_{i=1}^{4}\alpha_k^2(i)T_k^2 + \frac{1}{4}\sum_{i=1}^{4}\sum_{j=1}^{4^{k-1}}E_{k-1}^2(i,j) \\ &= \frac{1}{4}\sum_{i=1}^{4}\alpha_k^2(i)T_k^2 + \frac{1}{4}\sum_{i=1}^{4}\mathrm{MSE}_{k-1}(i).\end{aligned} \tag{39}$$

In the worst case $|\ \alpha_k(i)\ | = 1$ and $\mathrm{MSE}_{k-1} = T_1^2 + T_2^2 + \cdots + T_{k-1}^2$. Thus,

$$\mathrm{MSE}_k \le \sum_{i=1}^{i=k} T_i^2. \tag{40}$$

Q.E.D.

REFERENCES

[1] R. G. Gallager, *Information Theory and Reliable Communications.* New York: Wiley, 1968.

[2] T. Berger, *Rate Distortion Theory: A Mathematical Basis for Data Compressions.* Englewood Cliffs, NJ: Prentice-Hill, 1971.

[3] A. J. Viterby and J. K. Omura, *Principles of Digital Communication and Coding.* New York: McGraw-Hill, 1979.

[4] S. P. Lloyd, "Least squares quantization in PCM," *Bell Labs Tech. Note* (1957) (Honorary publication in the *IEEE Trans. Inform. Theory,* vol. IT-28, pp. 129–137, Mar. 1982).

[5] J. Max, "Quantizing for minimum distortion," *IRE Trans. Inform. Theory,* vol. 6, 1960.

[6] A. Segal, "Bit allocation and encoding for vector sources," *IEEE Trans. Inform. Theory,* vol. IT-22, no. 2, Mar. 1976.

[7] S. L. Tanimoto and T. Pavlidis, "A hierarchical data structure for picture processing," *Comput. Graphics, Image Processing,* vol. 4, 1975.

[8] A. Klinger and C. R. Dyer, "Experiments on picture representation using regular decomposition," *Computer Graphics, Image Processing,* vol. 5, 1976.
[9] Y. I. Grosky and R. Jain, "Optimal quadtrees for image segments," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. PAMI-5, 1983.
[10] Y. Cohen, M. S. Landy, and M. Pavel, "A hierarchical coding of binary images," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. PAMI-7, 1985.
[11] C. H. Chien and J. K. Aggarwal, "A normalized quadtree representation," *Comput. Vision, Graphics, Image Processing,* vol. 26, 1984.
[12] H. Samet and M. Tamminen, "Computing geometric properties of images represented by linear quadtrees," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. PAMI-7, no. 2, 1985.
[13] P. M. Farrelle, *Recursive Block Coding for Image Data Compression.* New York: Springer-Verlag, 1990, ch. 6, 7.
[14] R. Wilson, "Quad-tree predictive coding: a new class of image data compression algorithms," in *Proc. Int. Conf. Acoust., Speech, Signal Processing,* 1984, pp. 29.3.1–29.3.4.
[15] C. A. Shaffer and H. Samet, "Optimal quadtree construction algorithms," *Comput. Vision, Graphics, Image Processing,* vol. 37, 1987.
[16] A. Hunter and P. Willis, "Classification of quad-encoding techniques," *Comput. Graphics Forum,* vol. 10, 1991.
[17] P. Strobach, "Quadtree-structured interframe coding of HDTV sequences," in *Proc. SPIE Int. Conf. Visual Commun., Image Processing* (Cambridge, MA), Nov. 1988.
[18] P. Strobach, "Quadtree-structured linear prediction models for image sequence processing," *IEEE Trans. Pattern Anal.,, Machine Intell.,* vol. 11, no. 7, July 1989.
[19] P. Strobach, "Tree-structured scene adaptive coder," *IEEE Trans. Commun.,* vol. 38, no. 4, Apr. 1990.
[20] P. Strobach, "Image coding based on quadtree-structured recursive least-squares approximation," in *Proc. Int. Conf. Acoust., Speech, Signal Processing,* 1989.
[21] P. Strobach, "A computation of quadtree rate distortion functions," *IEEE Trans. Inform. Theory,* to be published.
[22] M. E. Blain and T. R. Fischer, "A comparison of VQ techniques," *EURASIP Image Commun.,* vol. 3, no. 1, Feb. 1991.

E. Shusterman received the B.Sc. and M.Sc. degrees in electrical engineering (cum laude) from Tel-Aviv University, Tel-Aviv, Israel, in 1983 and 1988, respectively. He is currently working toward the Ph.D. degree in electrical engineering.

From 1983 to 1988, he was with the Israeli Aircraft Industry as an Electronics Design Engineer. From 1988 to 1990, he was with Dazix Systems, Israel. His research interests include image compression and signal processing.

M. Feder (S'85–M'87) received the B.Sc. and M.Sc. degrees (summa cum laude) from Tel-Aviv University, Tel-Aviv, Israel, and the Sc.D. degree from the Massachusetts Institute of Technology, Cambridge, and the Woods Hole Oceanographic Institution (WHOI), Woods Hole, MA, all in electrical engineering, in 1980, 1984, and 1987, respectively.

During 1987–1988, he was a Research Associate and Lecturer in the Department of Electrical Engineering and Computer Science at M.I.T. He then was with Elbit Computers, Haifa, Israel, in signal processing and image compression. In October 1989, he joined the faculty of the Department of Electrical Engineering—Systems, Tel-Aviv University. He was also a Visiting and Guest Investigator at WHOI in the summers of 1983 and 1988–1991, and a Visiting Scientist at Scripps Institute of Oceanography, University of California–San Diego, in the summer of 1992. His research interests include data and signal compression, topics in information theory, signal and image processing, and sonar signal processing. Since June 1993 he has been an Associate Editor for Source Coding for the IEEE TRANSACTIONS ON INFORMATION THEORY.