Image Compression Using Binary Space Partitioning Trees

Hayder Radha, *Member, IEEE,* Martin Vetterli, *Fellow, IEEE,* and Riccardo Leonardi, *Member, IEEE*

Abstract— **For low bit-rate compression applications, segmentation-based coding methods provide, in general, high compression ratios when compared with traditional (e.g., transform and subband) coding approaches. In this paper, we present a new segmentation-based image coding method that divides the desired image using binary space partitioning (BSP). The BSP approach partitions the desired image recursively by arbitrarily oriented lines in a hierarchical manner. This recursive partitioning generates a binary tree, which is referred to as the *BSP-tree representation* of the desired image. The most critical aspect of the BSP-tree method is the criterion used to select the partitioning lines of the BSP tree representation. In previous works, we developed novel methods for selecting the BSP-tree lines, and showed that the BSP approach provides efficient segmentation of images. In this paper, we describe a hierarchical approach for coding the partitioning lines of the BSP-tree representation. We also show that the image signal within the different regions (resulting from the recursive partitioning) can be represented using low-order polynomials. Furthermore, we employ an optimum pruning algorithm to minimize the bit rate of the BSP tree representation (for a given budget constraint) while minimizing distortion. Simulation results and comparisons with other compression methods are also presented.**

I. Introduction

THE field of image compression is rich in diverse source coding schemes ranging from classical lossless techniques and popular transform approaches to the more recent segmentation-based (or second generation) coding methods [9], [12], [16]–[18], [20]. For low bit-rate compression applications, segmentation-based coding methods provide, in general, high compression ratios when compared with traditional transform, vector quantization (VQ), and subband (SB) coding approaches.

The notion of segmentation-based image coding was introduced during the early 1980's [11], [12]. A contour-texture coding method was used to partition the image into rather complex geometric regions. The image signals within these regions were represented using low-order polynomials. Quadtree-based image compression, which recursively divides the image signal into simple geometric regions (squares or rectangles), has been one of the most popular segmentation-based coding schemes investigated by researchers [13], [14], [30], [31]).

Segmentation-based compression methods usually describe the desired image as a set of regions. In general, the description of each region requires two types of information: 1) the geometry of the region boundaries and 2) the attributes of the image signal within the region. In order to achieve high compression ratio and good image quality, one needs to segment the image into a minimum number of regions such that the geometric description of the regions' boundaries is simple and the image signal within each region is continuous (or smooth).

Therefore, the most challenging aspect of a segmentation-based coding approach is to balance between a small number of geometrically simple regions and the smoothness (or continuity) of the image signal within these regions. Previous segmentation-based methods have focused on one of these two conflicting requirements at the expense of the other. For example, due to the complexity of the regions used in the contour-texture coding method [12], the geometric information uses 75% of the total bit-rate cost even with a very small number of regions. On the other hand, using rigid geometric descriptions, such as the quadtree representation method, a very large number of regions is needed to generate a piecewise-smooth image signal [13], [14].

The binary tree image coding method presented here achieves the above balance by using a simple, yet flexible, description of the image regions. Due to this flexible geometric description, which is based on arbitrarily oriented lines, a small number of regions can be used to represent the image while maintaining simplicity for the regions' boundaries, and smoothness for the image signal within these regions.

This paper is organized as follows: In the next subsection we provide an overview of binary tree representations of images (based on our work and other related work). Then, we describe our binary-tree-based image compression method in Section II. The line quantization and coding method, partitioning line selection criterion, and the coding of the attributes of the image signal (using first-order approximations) are described in Sections II-A, II-B, and II-C, respectively. In Section III, we develop an optimum pruning algorithm based on the generalized Breiman, Friedman, Olshen, and Stone (G-BFOS) method [2] to reduce the bit rate of the binary tree representation to a required budget constraint while minimizing distortion.

Manuscript received September 2, 1994; revised January 3, 1996. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Michael T. Orchard.

H. Radha is with Philips Research Laboratories, Briarcliff Manor, NY 10510 USA (e-mail: hmr@philabs.research.philips.com).

M. Vetterli is with the Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94720-1770. USA

R. Leonardi is with the Signals and Communications Laboratory, Department of Electronics for Automation, University of Brescia, Italy.

Publisher Item Identifier S 1057-7149(96)08495-3.

Fig. 1. Examples of image segmentation using binary space partitioning.

Finally, in Section IV, we show simulation results and compare our work with other image compression methods.

A. Binary Tree Representation of Images

A binary tree image representation method that divides the image domain using binary space partitioning (BSP) was introduced by the authors in 1990 [22] and further developed in a series of papers [23]–[25]. The BSP approach partitions the desired image recursively by straight lines in a hierarchical manner. First, a line (arbitrarily oriented) is selected, based on an appropriate criterion, to partition the whole image into two subimages. Using the same (or another) criterion, two lines are selected to split the two subimages resulting from the first partitioning. This procedure is repeated until a terminating criterion is reached. The outcome of this recursive partitioning is a set of (unpartitioned) convex regions (polygons) that are referred to as the cells of the segmented image (see Fig. 1), and a binary tree that is referred to as the *binary space partitioning tree* representation of that image. The nonleaf nodes of the BSP tree are associated with the partitioning lines, and the leaves represent the cells (unpartitioned regions) of the image. Each region of the image's BSP tree representation may have one or more attributes that describe the characteristics of the image signal. An example of such attributes is a zero-order (i.e., the mean value) or higher order polynomial model of the pixel intensities within the region.

The most critical aspect of the BSP tree construction process is the method used for selecting the partitioning lines. At each stage of the recursive partitioning (including the first stage when partitioning the whole image), the BSP line-selection method consists of two major steps, explained in the following:

1) Since the number of lines that pass through (i.e., partition) the polygon under consideration is infinite, in practice, and especially for image coding applications, one has to *quantize* the space of all possible lines that partition the polygon. This *line-quantization* process generates a finite set of lines that one needs to consider. Furthermore, because lines in the image (two-dimensional) domain can be represented using two parameters (e.g., the slope-intercept or normal[1] representation), a line-quantization process implies the quantization of two parameters. In general, and independent of the particular line representation used, one of the two line parameters is referred to as the *line orientation*.[2] The quantization of the line orientation is an important concept and will be discussed further below.
2) From the finite set of (quantized) lines, one has to select *the* partitioning line of the polygon under consideration. For this step, one needs to define a *line-selection criterion* that can be used for selecting the partitioning line.

In earlier stages of our work [22], [24], the number of line-orientations we considered (at every step of the recursive BSP tree construction process) was on the order of the image dimension[3] (e.g., 256 orientations for 256 × 256 images). In [23], we described a multiresolution approach for constructing the BSP tree representation of images. Under this approach, low-resolution images (e.g., with 32 × 32 pixels) were derived from higher resolution images, and were used to construct a low-resolution BSP tree. This enabled us to consider a small number of line orientations when building the low-resolution BSP tree (see [23] and [24] for more details).

The binary-tree-based coding scheme that is most related to the BSP tree method is the adaptive tree-structured segmentation (ATSS) approach introduced by Wu in 1992 [34]. (The two methods were developed independently.) Similar to the BSP scheme, the ATSS method i) segments the image in a binary recursive manner using straight lines and ii) represents the image using a binary tree data structure. The main difference, however, is that the ATSS method restricts the partitioning to lines that have one of *four* possible orientations: 0°, 45°, 90°, and 135°. Therefore, the ATSS approach quantizes the line orientation into four values, uniformly, at all stages of the recursive partitioning. The ATSS-based segmentation leads to polygons with a maximum of eight vertices, whereas the BSP-tree approach gives arbitrarily shaped (convex) polygons with variable number of vertices. In other words, the BSP tree representation approach provides a more general framework for the recursive binary partitioning of images. By using more line orientations, the BSP tree approach can represent complex images with fewer regions. This leads to better quality images at a given bit rate (see the simulation section).

Regarding the line-criterion used for selecting the partitioning lines, in our previous works we have employed both *boundary* and *least-square-error*-based criteria [26]. Under the boundary-based approach, the selected lines pass through the boundaries (edges) of the objects in the image. For the least-square-error case, a selected line minimizes a square error function defined over the region under consideration. The ATSS approach employs a least-square-error criterion for selecting the partitioning lines [34].

In this paper we present a hierarchical, least-square-error-based (LSE-based) BSP-tree image compression method. Un-

[1] The slope-intercept, $y = mx + b$, representation used the (m, b) parameters, whereas the normal, $\rho = x \cos \theta + y \sin \theta$, representation uses the (θ, ρ) parameters. The ρ parameter represents the normal distance between the line and the origin $(x = 0, y = 0)$. And the θ parameter is the angle between the line normal and the x-axis.

[2] The line orientation is represented by the slope (m) and angle (θ) parameters of the slope-intercept and normal line representations, respectively.

[3] Although this quantization may be costly (in terms of bits) from an image coding perspective, it is consistent with the normal practice used in computer vision applications for detecting straight lines in edge images [7].

der this approach, a) quantization of the line orientations is done in a hierarchical manner, i.e., the number of lines considered (for partitioning a polygon) is proportional to the size of the polygon, and each selected partitioning line meets a square-error criterion.

II. Hierarchical LSE-Based Image Coding Using BSP Trees

As mentioned above, at each step of the recursive partitioning of the BSP-tree coding method, one has to identify a finite number of lines that need to be considered for segmenting the region under consideration. Only one line from this finite set is selected (using the LSE criterion described in Section II-B) to partition the region into two subregions. Under the hierarchical line quantization approach used here, the number of line *orientations* considered depends on the dimension of the polygon as described below. In addition, we use *lossless* coding of the (quantized) partitioning line parameters.

Both the terminating criterion, used to terminate the recursive partitioning and create the unpartitioned convex polygons, and the line selection criterion, used to select the partitioning line from the quantized set of lines, are based on a least-square-error approach. When the mean square error (MSE) resulting from approximating the image signal (using a first-order polynomial) is less than a desired threshold, then the region under consideration is not partitioned (i.e., BSP-tree cell). In this case, the polynomial parameters (cell attributes) are quantized and coded. In our work, we use both uniform and hierarchical quantization methods for the first-order polynomial parameters as described in Section II-C.

If the MSE error is higher than the desired threshold, one of the quantized lines is selected to partition the region under consideration. The selected line has to minimize a square-error function defined over that region. Since the number of lines considered for partitioning the whole image (as well as the large polygons) is very large,[4] we employed the LSE partitioning line (LPL) transform [25], which significantly reduces the number of quantized lines that need to be considered (see Section II-B).

When coding the BSP-tree representation, one bit is used to indicate whether the current polygon is partitioned or not. We refer to this bit as the *binary tree structure bit* since it describes the shape and size of the BSP tree. Fig. 2 shows an example of a BSP tree and its data stream. Therefore, the encoded *BSP tree data stream* consists of the i) binary tree structure bits; ii) encoded line parameter bits (the L_1, L_2, and L_3 data shown in Fig. 2; and iii) coded bits (the C_1, C_2, C_3, and C_4 data shown in the same figure) of the quantized polynomial parameters approximating the image signal in the unpartitioned regions.

A. Quantization and Coding of the Line Parameters

As mentioned above, quantization of the lines implies, in general, the quantization of the parameters used to represent these lines. There are several parametric representations of straight lines in the 2-D plane. Here we focus on the *normal* representation (or the (θ, ρ) parameterization) since it simplifies the line detection process when compared with the *slope-intercept* representation [4]. Therefore, the remainder of this section focuses on the quantization of the (θ, ρ) parameters. First, we take a closer look at the (θ, ρ) parameter space that we are about to quantize.

A point in the (θ, ρ) parameter space represents a straight line in the image space (2-D plane). From the dimensions of the image domain, one can exactly identify the set of all lines that can pass through (or partition) the domain. This set of partitioning lines corresponds to (or is represented by) a set of *points* in the parameter space. We refer to this set of points as the *partitioning lines domain* (PLD).

The PLD domain is usually a connected region in the parameter space as shown in Fig. 3. The boundaries of the

[4] On the order of the number of pixels in the image.

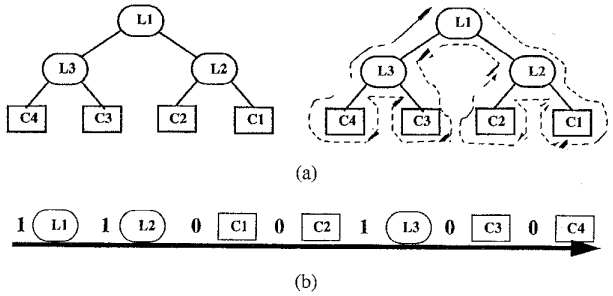


Fig. 2. (a) BSP tree. (b) Traversal of the tree. (c) BSP-tree data stream resulting from the tree traversal.

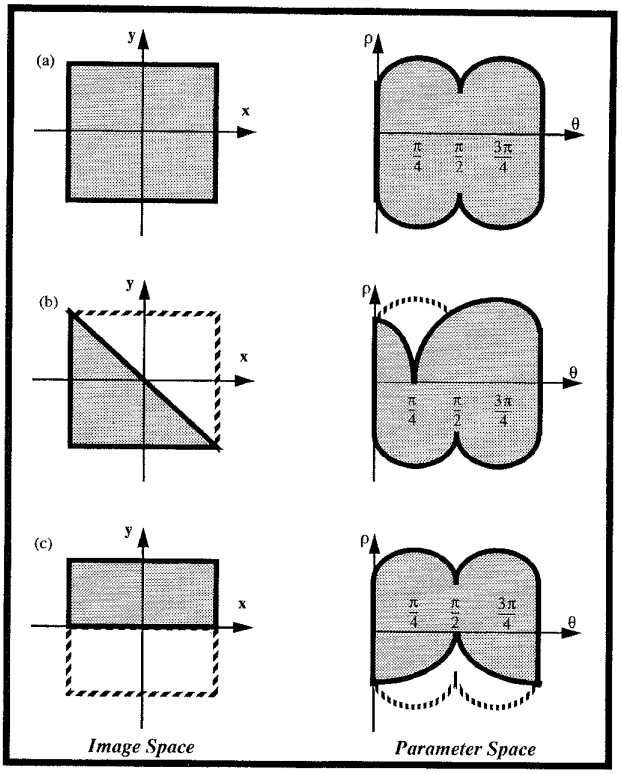


Fig. 3. Different regions in the image space and the corresponding PLD domains in the parameter space. (a) Entire image and its PLD domain. (b) and (c) Two subregions of the image and their PLD domains.

PLD domain of an image are defined by the parametric representation of the image-domain four corner points. For example, if the image domain has the corner points (x, y) = (1, 1), $(-1, 1)$, $(-1, -1)$, and $(1, -1)$, then these points are mapped onto four sinusoidal functions in the parameter space using $\rho = x \cdot \cos\theta + y \cdot \sin\theta$. Now, for a given value of θ, the lower and upper limits of the PLD domain are determined by the minimum and maximum values of the four parametric (sinusoidal) curves generated by the four corner points of the image.

Similarly, knowing the geometric description of a given polygon, one can identify the parameter space domain of all possible lines that can pass through this polygon. In other words, each polygon in the image domain has a corresponding *unique* PLD domain in the parameter space.

We assume that the decoder will have *a priori* knowledge of the image dimensions (otherwise, the dimensions are sent first). Hence, the first partitioning line representation in the parameter space has to be within the PLD domain of the image. Due to the recursive nature of the BSP-tree representation, the BSP-tree decoder (at the receiver) has a complete geometric description of the polygon to be partitioned. Consequently, the receiver has a knowledge of the PLD domain of that polygon, and expects the parametric representation of the partitioning line to be within that PLD domain. As the recursive partitioning continues, the polygons get smaller and so do their corresponding PLD domains in the parameter space. Fig. 3 shows examples of polygons and their PLD domains in the (θ, ρ) space.

Now, let's consider the quantization aspects of the PLD domain. Quantizing straight lines, which pass through the discrete 2-D lattice of pixels, is a very difficult and complex problem. The line quantization process is influenced by how one defines a pixel and its domain in the 2-D (x, y) space. For example, a pixel at location (x_0, y_0) can be viewed as a square centered at (x_0, y_0), and its value represents the image intensity over that (whole) square. Another example is that a pixel represents just a single point in the (x, y) space and its value represents the image signal only at that point. In addition to the pixel definition, when quantizing the lines that "pass through" (whatever that means) the image pixels, one has to consider the line width. For example, if a pixel is defined as a unit square (i.e., has a 1×1 dimension), should the line width be the unit dimension (i.e., one), the square diagonal (i.e., $\sqrt{2}$), or dependent on the line orientation?

These and other tricky questions forced us to use a simple and heuristic approach for the quantization of lines that pass through a convex polygon. We think of pixels as discrete points in the 2-D (x, y) space with quantization step sizes of one in both the horizontal (x) and vertical (y) directions. Our objective, in general, is to consider lines that "pass through" different set of pixels (Fig. 4). Consider the case when one line passes through a set of pixels and another line passes in the "vicinity" of the same set of pixels. Now, if the second line does not impact (i.e., neither passes through nor passes in the vicinity of) any other pixels outside the aforementioned set, then the second line is discarded. However, *if* one can define the set of all discrete lines such that each line in the set

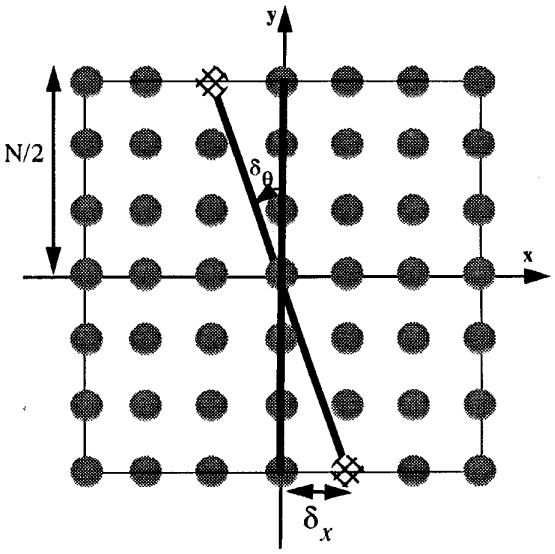


Fig. 4. Quantization size of the angle θ when the normal distance ρ is zero.

"pass through" or "pass in the vicinity of" a "different" set of pixels, we can not claim that our heuristic approach captures that entire set of lines. (This depends on how one defines "pass through," "vicinity," and "different.") Without having much choice, when developing our line quantization method we resorted to the following principles, which we believe are reasonable.

1) The number of line orientations (i.e., discrete θ values) should be dependent on the polygon size.
2) The range of parallel lines (i.e., range of the ρ parameter) that can partition the polygon under consideration should be dependent on the polygon size and the lines' orientation.
3) The quantization step size of the ρ parameter should be a function of the line orientation.

To further simplify the line quantization process, we perform the following: i) we code the θ and ρ parameters separately (not jointly); ii) for a given polygon, we use uniform quantization of the line orientations; and iii) for a given orientation, we use uniform quantization of the ρ parameter. Below, we justify, through simple examples, the above principles, and explain our quantization and coding method for the θ and ρ parameters. Before proceeding, it is important to note the following: Since the quantization and coding of the ρ parameter is dependent on the line orientation (θ value), under our proposed line coding approach the encoder must first encode and transmit the parameter θ and then the parameter ρ.

1) Quantization and Coding of the Line Orientation Θ*:* Regardless of the type of line-orientation quantization used, one has to consider the range between zero and π for the angle θ. This range of θ values is also independent of the size of the polygon. We show here, through an example, that the quantization step size for θ (i.e., δ_θ) should be *dependent* on the size of the polygons. For large polygons one needs to use small quantization levels, whereas smaller polygons can tolerate large quantization levels.

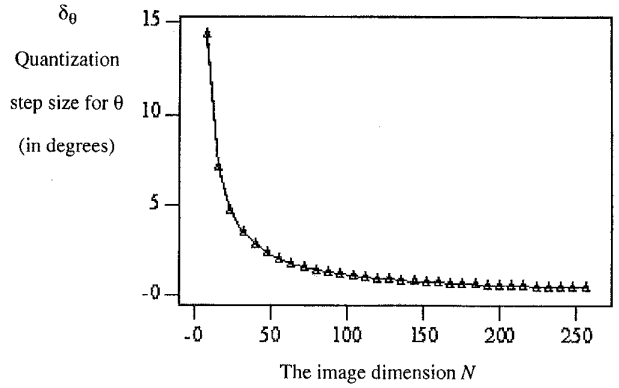


Fig. 5. Plot of δ_θ as a function of the image dimension N. Also shown are the values of the approximation $\delta_\theta = 1/(N/2)$. The approximation values, which are shown with the symbol Δ, illustrates that the quantization step size δ_θ one employs should be inversely proportional to the image dimension.

Let us consider the example shown in Fig. 4, where the image domain (with a dimension N) is centered around the origin. First, consider the vertical line $(\theta, \rho) = (0, 0)$. This line passes through all pixels with *x-coordinates* equal to zero. If we rotate the line around the origin while maintaining the ρ value constant (i.e., $\rho = 0$), then the first pixels that intersect the rotated line are located at $(x, y) = (-1, N/2)$ and $(x, y) = (1, -N/2)$ If one uses this rotated angle as the quantization step size for θ, then $\delta_\theta = \text{atan}\,[2/N]$.

Fig. 5 shows a plot for this δ_θ as a function of the image dimension N. As the image dimension gets larger, the value for δ_θ gets smaller. In other words, one needs to use larger number of θ values for larger images. This simple example illustrates that changing the line orientation by a small angle, which is inversely proportional to the image dimension, causes the rotated line to cover a new set of pixels. This observation is the basis for our proposed hierarchical line-orientation quantization approach.

Similar to the image case, the quantization step size δ_θ required for a given polygon is also a function of the polygon dimensions. In this case, however, it is impractical (from analysis and implementation point-of-view) to design a θ quantizer that is a function of a polygon with an arbitrary (convex) shape. One way of solving this problem is to employ a *bounding box* around the polygon under consideration. Now, the quantization step size δ_θ is dependent on the dimension of the box surrounding the polygon, similar to the case of an image with a dimension N. This strategy can be adopted since the decoder has a complete knowledge of the geometry of the polygon, and therefore can derive the dimensions of the bounding box for that polygon. In general, the bounding box is an $M \times N$ rectangle. We adopted the following strategy when quantizing the partitioning lines of a polygon: The number of line orientations (num_θ) considered is proportional to the polygon's bounding box maximum dimension. In other words $num_\theta \propto \max\{M, N\}$.

To further simplify the quantization and coding of the line orientation, we use, for all polygons with the same $M \times N$ (or $N \times M$) bounding box, a *fixed* number of bits for a *uniform* and *lossless* coding of the (quantized) θ values.[5] In our line quantization approach, for the polygon under consideration with an $M \times N$ bounding box, we start by determining the (maximum) number of line orientations using the following formula: $\text{num}_\theta = \lceil \max\{M, N\} \rceil_2$, where $\lceil f \rceil_2$ is the smallest power-of-two integer that is larger than f. Therefore, the (uniform) quantization step size can be expressed as $\delta_\theta = \pi/\lceil \max\{M, N\} \rceil_2$. Now, the quantized θ values are $\theta_i = i\delta_\theta = i(\pi/\lceil \max\{M, N\} \rceil_2)$, for $i = 0, 1, 2, \cdots, \text{num}_\theta - 1$. Consequently, the number of bits $\Re_\theta$ used for lossless coding of the (quantized) angle θ of a partitioning line is $\Re = \log_2 [\text{num}_\theta] = \log_2 [\lceil \max\{M, N\} \rceil_2]$.

2) Quantization and Coding of the ρ Parameter: As explained above, the reduction in the PLD domains (due to smaller polygons) is achieved merely by limiting the *range* of ρ values. For a given polygon P (see Fig. 3), the range of ρ values within the PLD domain of P is a function of the angle θ. If one defines the ρ range by the interval $[\rho_{\min}, \rho_{\max}]$, then $\rho_{\min}$ and $\rho_{\max}$ can be computed as follows:

$$\rho_{\min}(\theta_i) = \min_{(x,y)\in V} \{x \cos\theta_i + y \sin\theta_i\}$$
$$\rho_{\max}(\theta_i) = \max_{(x,y)\in V} \{x \cos\theta_i + y \sin\theta_i\} \qquad (2.1)$$

where V is the set of vertices of the polygon P and θ_i is the quantized line-orientation value. Now, the range of ρ values is $\rho_{range}(\theta_i) = \rho_{\max}(\theta_i) - \rho_{\min}(\theta_i)$.

In addition to the range value, the number of bits required to code ρ is dependent on the size of its (ρ) *quantization levels*. We show here, through a simple example, that the size of the quantization levels for ρ should be a function of the angle θ. First, consider the horizontal line $(\theta = \pi/2, \rho = 0)$ that passes through all pixels with a y-coordinate equal to zero. If we increase the ρ value of this line (while keeping its line orientation $\theta = \pi/2$ constant) then the new set of pixels affected by the moving line are the ones with y-coordinate equal to one. This new line is represented by the $\theta = \pi/2, \rho = 1$ parameters. Now, if we start with the diagonal line $(\theta = \pi/4, \rho = 0)$, and increase its ρ value (while keeping its line orientation $\theta = \pi/4$ constant), then the new line that passes through a new set of pixels is represented by the $\theta = \pi/4, \rho = 1/\sqrt{2}$ parameters. Therefore, the quantization step sizes for the horizontal and diagonal lines are 1 and $1/\sqrt{2}$, respectively. This simple example illustrates that the ρ quantization levels used should be a function of the line orientation. We use the following expression for a θ-dependent quantization of ρ:

$$\delta_\rho(\theta_i) = \max\{|\cos\theta_i|, |\sin\theta_i|\} \qquad (2.2)$$

where θ_i is the quantized line orientation value. It is important to note that, for a given value of θ_i, $\delta_\rho(\theta)$, represents a *uniform* quantizer. Therefore, the quantized ρ values can be expressed as follows:

$$\rho_j = j\delta_\rho(\theta_i) = j[\max\{|\cos\theta_i|, |\sin(\theta_i|\}] \quad \text{for } j = \min_j, \cdots, \max_j \qquad (2.3)$$

[5] In other words, we do not employ entropy-based coding for the line orientation. The design of an optimum variable-length code (VLC) which is dependent on the polygon dimension is difficult and requires further study.

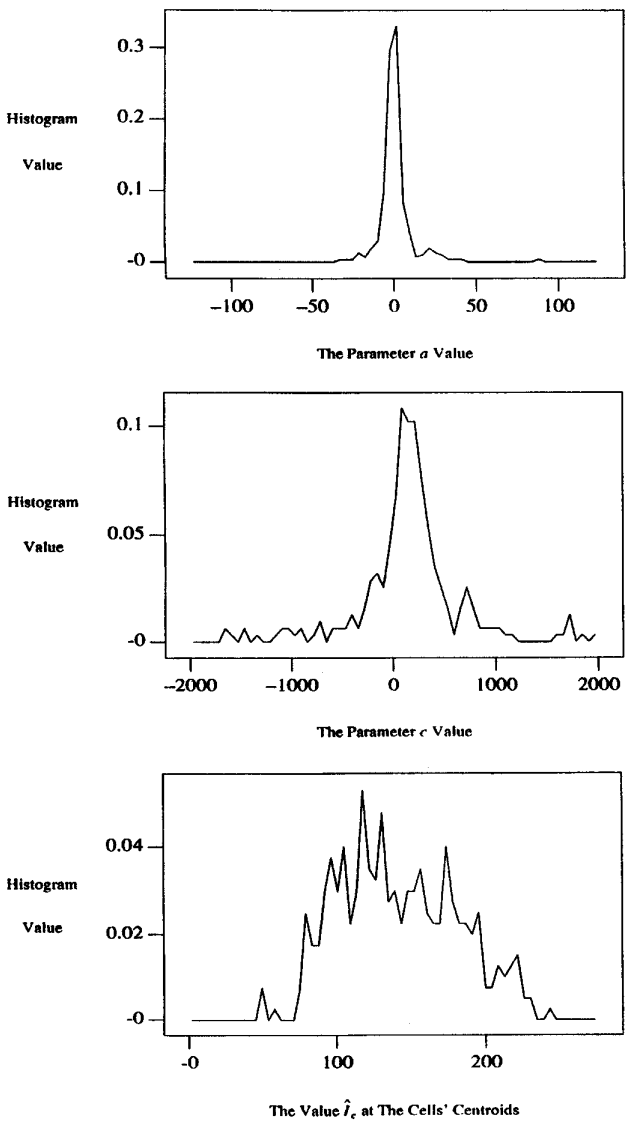


Fig. 6. Histogram functions of the first-order polynomial $(ax + by + c)$ coefficients for the Mona Lisa example. (a) Histogram of the coefficient a (similar histogram was obtrained for the coefficient b). (b) Histogram of the c parameter $\hat{I}_c$ in the approximation $\hat{I} = a(x - x_c) + b(y - y_c) + \hat{I}_c$, where (x_c, y_c) is the coordinate of the polygon centroid. By comparing the histograms in (b) and (c), one can see that the parameter $\hat{I}_c$ has a much smaller entropy than the DC parameter c in the first-order polynomial $ax + by + c$.

where $\min_j = \lceil \rho_{\min}(\theta_i)/\delta_\rho(\theta_i) \rceil$ and $\max_j = \lfloor \rho_{\max}(\theta_i)/\delta_\rho(\theta_i) \rfloor$. (Here $\lceil f \rceil$ and $\lfloor f \rfloor$ are the smallest and largest integers that are larger and smaller than f, respectively.)

Based on this approach, the number of quantized ρ values used for a given (quantized) line orientation is: $\text{num}_\rho = \max_j - \min_j + 1$. Consequently, for a given polygon with a vertex set V, and a quantized angle θ_i of a line (that partitions the polygon), the number of bits $\Re_\rho$ used for lossless (and uniform) coding of the normal distance parameter ρ is $\Re_\rho = \lceil \log_2 [\text{num}_\rho] \rceil$.

B. The LSE-Based Line Selection Method

One has to select the partitioning line of a given region from the set of quantized lines (described above) based on some criterion. If Λ_P represents the set of quantized lines considered for partitioning a polygon P, then based on the θ and ρ quantization approach described above, Λ_P can be expressed as shown in (2.4) at the bottom of the page. Now, every line h_{ij} in the set Λ_P has to be tested before selecting the partitioning line. In a previous work [21], [25], we developed an optimization approach, in a (recursive) least-square-error (LSE) sense, for constructing BSP trees. Based on this LSE approach, a straight line is selected to partition a given region of the image if the line minimizes a square error function defined over that region. If $E(P; h_{ij})$ is the square error function resulting from partitioning the polygon P by the line h_{ij}, then the LSE (selected) line satisfies the following:

$$\hat{h}_{ij} = \min_{h_{ij} \in \Lambda_p} [E(P; h_{ij})].$$

See [25] for the definition of the function $E(P; h_{ij})$.

When large polygons (e.g., the whole image) are under consideration, evaluating the square error function for every line in the set Λ_P can be very computationally expensive. For an image with an $N \times N$ dimension, the computational complexity can be on the order of N^4. In [26], we derived a necessary condition for the optimum partitioning lines of a BSP tree representation. We used this necessary condition to develop an LSE partitioning line (LPL) transform. The LPL transform can reduce the number of lines that needs to be considered for the LSE error test significantly. The computational savings for an $N \times N$ image are on the order of N. In this work, the LPL transform is employed by the LSE-based approach to select the partitioning lines in a recursive manner when building the BSP tree of the desired image. For more details regarding the LPL transform and the LSE-based method for selecting the partitioning lines, the reader is referred to [25] and [26].

C. Coding the Cell Attributes

As explained in the introduction, one of the main advantages of segmentation-based coding methods is that the image signal within the unpartitioned regions can be represented using simple mathematical functions. For the BSP tree approach, the characteristics of the image signal within the unpartitioned regions are referred to as the *cell attributes*. We focus below on using polynomial functions to describe the cell attributes of a BSP tree representation. If the image signal $I(x, y)$ over a polygon domain D is approximated using a polynomial $\hat{I}$, then

$$\Lambda_P = \left\{ h_{ij} = (\theta_i, \rho_j): \begin{array}{ll} \theta_i = i\delta_\theta(P), & i = 0, 1, 2, \cdots, \text{num}_\theta(P) - 1 \\ \rho_j = j\delta_\rho(\theta_i), & j = \min_j(\theta_i), \cdots, \max_j(\theta_i) \end{array} \right. \tag{2.4}$$

the optimum, in the square-error sense, estimate $\hat{I}_0$ satisfies the following:

$$\hat{I}_0 = \min_{\hat{I}} \int_D [I(x,y) - \hat{I}]^2 \, dx \, dy. \tag{2.5}$$

The square-error distortion depends on both the smoothness of the image signal within the cell and the order of the polynomial. The higher the order and the smoother the image signal, the less distortion results from the approximation. Due to the efficient segmentation of the LSE-based method used here for selecting the partitioning lines, one can use low-order polynomials to describe the image signal within the cells. Here, we consider first-order polynomials for representing the image signal within the BSP-tree cells: $\hat{I} = ax + by + c$.

The optimum approximation $\hat{I}_0$ that minimizes the square error function is found by determining the coefficients a, b, and c that minimizes

$$\int_D [I(x,y) - (ax + by + c)]^2 \, dx \, dy. \tag{2.6}$$

After computing the polynomial coefficients, one needs to encode these coefficients efficiently. It is important to note that a and b measure how flat the image signal is along the x and y axes, and c indicates the average brightness of the image signal. The smaller the value of a and b, the flatter the image signal, and the larger the value of c the brighter the image signal.

1) Coding the Slope Parameters: Fig. 6(a) shows the histogram of the optimum coefficient a for a BSP tree representation of the Mona Lisa image. As expected, the small variation in the coefficient's values gives a clear indication of the smoothness of the image signal within the BSP tree cells. The average entropy of the quantized values of the slope parameters a and b is a function of the range of values considered and the number of quantization levels. Fig. 7 shows the entropy (in bits/coefficient) of the parameter a for the Lenna[6] BSP-tree example for different uniform quantizers. It is clear that for a given range value, the entropy increases with the number of quantization levels. Similarly, for a given number of quantization levels, the entropy, in general,[7] increases with the range.

To select the desired uniform quantizer, one needs to consider the performance (e.g., in terms of peak signal-to-noise ratio (PSNR), or subjectively) that can be achieved for a given coding rate. Fig. 8 shows the average number of dB's of PSNR one can gain per every bit used to encode the BSP-tree Lenna image using different uniform quantizers. All images were coded at a rate around 0.1 b/pixel. Although all of the compressed images have very similar data rate, one can gain as much as 1 dB of PSNR by selecting the appropriate quantizer. These and other examples have shown that, for a target rate of around 0.1 b/pixel, a uniform quantizer with 256–512 quantization levels and a range value around 64 provides good PSNR performance numbers when compared with other

[6]For the remainder of the document, we focus on the 512 × 512 Lenna image, which is the most common test image used in the compression literature.

[7]For small number of quantization levels (e.g., two), the entropy actually decreases with increasing values of the range.

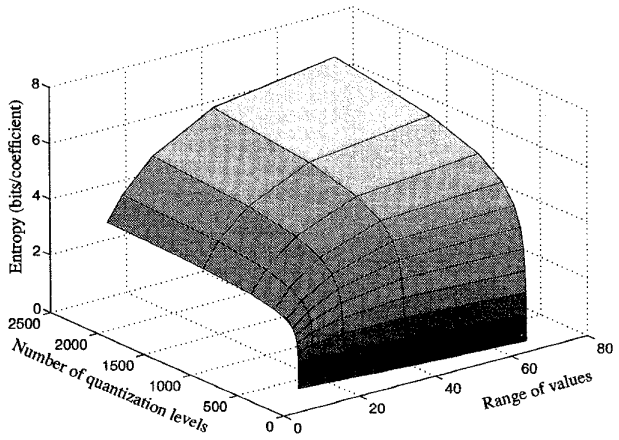


Fig. 7. Entropy (in bits per coefficient) of the coefficient a (for the Lenna BSP image example) as a function of the number of quantization levels and the range of values considered. Uniform quantization levels are used with a quantized value of zero included. For even number of quantization levels, the minimum and maximum values of the coefficient were chosen accoding to the following formulas: $min = \Delta_a((L-1)/2)$ and $max = \Delta_a((L-1)/2)$, where $\Delta_a = (range)/L$ is the quantization step size and L is the number of quantization levels. The corresponding minimum and maximum quantization values are $Q_{min} = min + (\Delta_a/2)$ and $Q_{max} = max - (\Delta_a/2)$.

uniform quantizers.[8] (Very similar results were obtained for the coefficient b.) For other target bit rates (e.g., 0.05 b/pixel), other uniform quantizers can provide better performance[9] as shown in the simulation section.

One major disadvantage of the (above) uniform quantizer is that it treats all polygons in the same manner. In other words, the number of quantization levels and range values are the same regardless of i) the number of pixels affected by this quantization and ii) the shape of the polygon. A better solution would be to have an adaptive (hierarchical) quantizer that uses different quantization levels and range values depending on the polygon under consideration. In this case, the number of quantization levels can be made directly proportional to the area of the polygon: $\text{num}_{levels} = (area_p)/(QF)$, where $area_P$ is the area of the polygon P and QF is a quantization factor for the slope parameters a and b. The number[10] of bits required to encode the coefficients a and b can be expressed as follows: $\Re_{slopes} = 2 \cdot \lceil \log_2 (\text{num}_{levels}) \rceil$.

[8]We have also considered encoding the slope parameters a and b in a differential manner. In this case, instead of encoding the value of the current slope, one encodes the difference between the current and previous slopes. No clear advantages for using this differential approach were observed. This indicates that the first-order attributes of *consecutive* cells in the BSP-tree data stream can be very different. However, we would like to emphasize that the result of differential coding depends heavily on the strategy used to define two *consecutive cells*. In our case, we computed the difference between the attributes of two consecutive cells within a BSP tree that are constructed using a preorder traversal. In other words, it is feasible that a more elaborate strategy for defining consecutive cells may result in a more efficient coding of the differential attributes.

[9]From our experience, at very low bit rates the PSNR performance measure does not provide a good criterion for comparing different quantizers (or even different encoders). In that case, visual evaluation might be the only tool available.

[10]By employing entropy-based coding for the different quantizers in the hierarchy, a smaller number of bits can be used. However, all the simulation results reported here are based on the maximum number of bits shown in the equation, and are therefore worst-case (or upper-bound) scenarios.

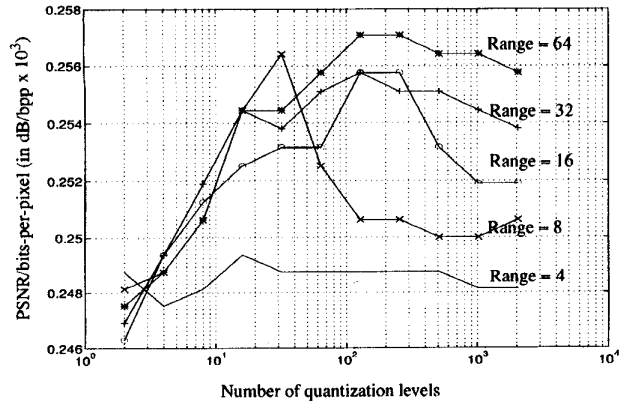


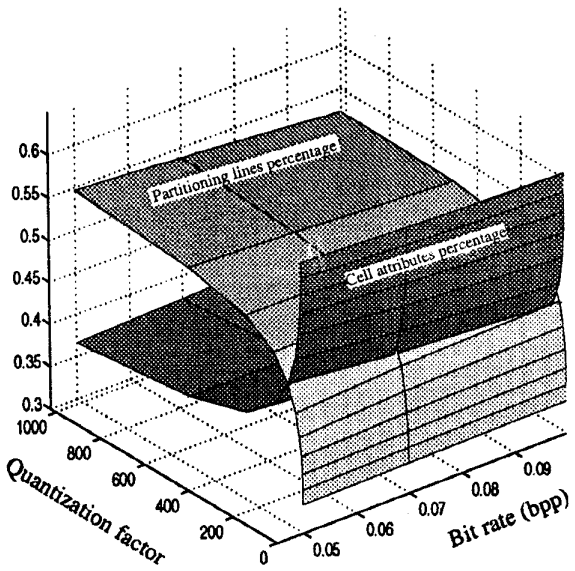
Fig. 8. The ratio of PSNR value to the average b/pixel for the Lenna example. The different points of the plots represents BSP tree images coded with different uniform quantizers for the parameters and of the first-order polynomial. The target bit rate was 0.1 b/pixel (the actual rates ranged from 0.097–0.1 b/pixel).

For a given bit rate, the quantization factor QF has an impact on the number of bits allocated for coding the cell attributes. Fig. 9(a) shows a plot for the percentage of bits allocated for the cell attributes and the partitioning lines as a function of QF and the bit rate for the BSP tree Lenna example. As QF increases the number of quantization levels for the slope parameters a and b decreases and this in turn decreases the number of bits[11] used to encode the cell attributes. Therefore, more bits are made available for coding the partitioning lines, which leads to an increase in the number of regions in the final image. Fig. 9(b) shows a similar plot for the number of cells as a function of QF and the bit rate. Moreover, the impact of the quantization factor on the overall performance of the adaptive quantizer depends on the desired bit rate. Fig. 10 shows a plot of the PSNR as a function of QF and the bit rate for the BSP tree Lenna example. As shown in the figure, the best QF value changes with the bit rate. For example, a QF of eight provides the best result (in terms of PSNR) when the target bit rate is around 0.1 b/pixel.

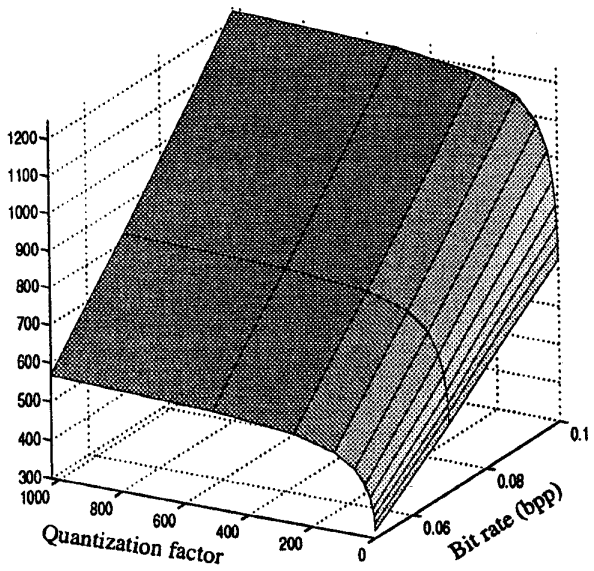
Similarly, one can change the range of values that should be considered based on the polygon dimensions. For a polygon P with a bounding box of dimensions $M \times N$, the maximum slope values (for the first-order approximation) that need to be considered are: $\max_a = (\max_{intensity})/M$ and $\max_b = (\max_{intensity})/N$, where $\max_{intensity}$ is the maximum pixel intensity value (i.e., 255 for 8 b/pixel images). Similar expressions can be used for the minimum slope values. Therefore, the total ranges of slope values that need to be considered by the quantizer are

$$range_a = \frac{2 \cdot \left(\max_{intensity} \right)}{M}$$

[11] Since a constant number of bits (always 8) are used to encode the DC parameter of the polynomial (as explained in the next section), the QF factor can be used to balance between the number of bits used on the segmentation aspect (i.e., partitioning lines) of the BSP tree representation and the number of bits used to represent the image signal.



(a)



(b)

Fig. 9. (a) Percentage of bits allocated for the cell attributes and the partitioning lines as a function of the quantization factor and the rate. (b) Number of cells (unpartitioned regions) as a function of the quantization factor and the bit rate.

and

$$range_b = \frac{2 \cdot \left(\max_{intensity} \right)}{N}. \tag{2.7}$$

As shown in the simulation results section, the hierarchical quantizer provides better performance when compared to the uniform quantizer.

2) Coding the Direct Current (DC) Parameter: The value of the coefficient c is strongly dependent on the location

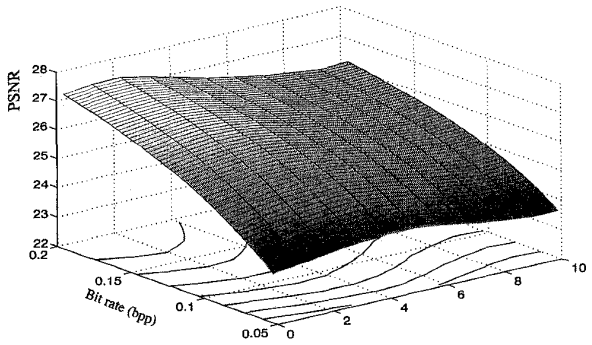


Fig. 10. PSNR as a function of the quantization factor (log) and the bit rate. As can be seen, the best quantization factor (in terms of maximum PSNR) changes with the desired bit rate.

of the cell's polygon relative to the origin of the image, i.e., c is *variant under translation*. In other words, for a given domain D of a BSP-tree cell, and an image function $I(x, y)$ over D, the value of c changes with the location of the domain D within the (x, y) plane. This property provides c with a wide range of possible values, and therefore makes c a bad choice for an efficient representation of the first-order polynomial. Fig. 6(b) shows the histogram of the coefficient c for the Mona Lisa BSP-tree example.

It is important to note that c represents the value of the first-order approximation at the origin, i.e., $\hat{I} = c$ at $(x, y) = (0, 0)$. Therefore, if the cell domain D does not include the origin point $(x, y) = (0, 0)$, c can have any value outside the normal range of a pixel luminance. This problem can be resolved by selecting a point (x_c, y_c) *within* the cell domain, evaluating the first order polynomial at that point, and using this value $\hat{I}(x_c, y_c)$ to represent the DC coefficient of the polynomial instead of the z-axis intercept c.

A good choice for the point (x_c, y_c) is the *centroid* of the cell domain D. In this case

$$x_c = \frac{\int_D x\, dx\, dy}{A}$$

and

$$y_c = \frac{\int_D y\, dx\, dy}{A} \tag{2.8}$$

where A is the area of the domain D. One advantage for using the centroid is that its coordinates x_c and y_c are computed when determining the optimum slope coefficients a and b. If $\hat{I}_c$ is the first-order polynomial value at the polygon centroid, then this polynomial can be expressed as follows:

$$\hat{I} = a(x - x_c) + b(y - y_c) + \hat{I}_c. \tag{2.9}$$

This expression can be obtained by translating the cell such that its centroid coincides with the origin $(x, y) = (0, 0)$. Now, the DC parameter $\hat{I}_c$ is invariant under translation. (In fact, $\hat{I}_c$ is equal to the mean value of i) the pixel intensities of the image signal over the region under consideration (i.e., the domain D), and ii) the first order approximation $\hat{I}$ over the same region.)

After receiving the values for a, b, and $\hat{I}_c$, a BSP-tree decoder needs to compute the centroid coordinates (x_c, y_c) in order to approximate the image signal within the cell under consideration (i.e., using (2.9)). This can be achieved since, as explained previously, a BSP-tree decoder will have a complete knowledge of the geometry of the polygon under consideration. Fig. 6(c) shows a plot of the histogram of the coefficient $\hat{I}_c$ for the Mona Lisa BSP-tree example. (Please note that different scales are used for Fig. 6(b) and (c).) By comparing this histogram with the plot of Fig. 6(b), it is clear that the parameter $\hat{I}_c$ has a much narrower range of values than the coefficient c. Since $\hat{I}_c$ has a value within the normal range of pixel intensities (i.e., 0–255), one can encode $\hat{I}_c$ efficiently using 8 bits per coefficient.[12]

III. Optimum Pruning of BSP Trees

In most coding applications, one is constrained by the usage of a certain number of bits to encode the desired signal. This is commonly known as a bit *rate budget constraint* for the encoder. After using an efficient compression strategy to code the signal, one way to reduce the bit rate (in order to meet a given rate budget constraint) is to reduce the amount of encoded data transmitted. In general, reducing the amount of data introduces some distortion to the encoded signal. The challenge in this case is to eliminate some data from the signal to achieve the maximum allowed bit rate such that the introduced distortion is minimum.

For a given BSP tree, reducing the number of bits required to represent the tree can be accomplished by *pruning* one or more branches of the original tree. Pruning a branch is basically converting an intermediate (or nonleaf) node into a terminating (or leaf) node. A pruned node can be thought of as the root node of a subtree of the original BSP tree. Since there are a large number of intermediate nodes in an encoded BSP tree (in the range of about 500–1500 nodes for the examples shown in the simulation section), one should have a well-defined strategy for selecting the nonleaf nodes, to be pruned, in order to achieve a given rate budget.

Here, we adopt the rate-distortion strategy, which guarantees a minimum distortion introduced to the signal when reducing the rate to the required budget. The same strategy guarantees the dual option of achieving a minimum bit rate for a maximum allowable distortion. Applying this strategy to a tree-structured data is known as *optimum* pruning. It is important to note, however, that the BSP tree representation method used here do not provide the optimum representation in a global rate-distortion sense. Finding the optimum BSP tree requires, in general, considering all possible combinations of i) recursive binary partitionings, ii) quantization and coding methods for the partitioning lines, and iii) quantization and coding methods for the image signal within the segmented

[12] It is feasible that a more efficient coding (e.g., 6 or 7 bits per coefficient) can be achieved for $\hat{I}_c$ by using entropy-based coding. The simulation results shown in this paper is based on an 8-bits/coefficient coding of the parameter $\hat{I}_c$, and therefore represent a worst case.

regions. The computational cost of finding such a tree is obviously very prohibitive. The main objective of employing the optimum pruning scheme described below is to adjust the bit rate of a *given* BSP tree representation in an optimum manner.

Optimum pruning has been used extensively, with great success, in other tree-based signal coding applications such as tree-structured vector quantization (TSVQ), quadtree encoding, and wavelet packet-based compression [2], [5], [27], [30]. Here, we use the G-BFOS optimum pruning algorithm developed in [2] for TSVQ applications.

Before describing how to apply the G-BFOS algorithm on a BSP tree representation of an image, we first present a formal definition of the problem. Let $\mathcal{R}(\mathcal{T})$ and $\mathcal{D}(\mathcal{T})$ be the rate and distortion of a BSP tree $\mathcal{T}$ representing the desired image. If $\mathcal{S}$ is a pruned version of the original tree $\mathcal{T}$, then $\mathcal{S}$ is a subtree of $\mathcal{T}$. Here we consider the subtrees of $\mathcal{T}$ that have the same root node t_0, where t_0 is the root node of the original tree $\mathcal{T}$. We use the notation $\mathcal{T} \Rightarrow \mathcal{S}$ to denote that $\mathcal{S}$ is a subtree of $\mathcal{T}$, and that both trees ($\mathcal{T}$ and $\mathcal{S}$) share the same root node. Let $\mathcal{R}(\mathcal{S})$ and $\mathcal{D}(\mathcal{S})$ be the rate and distortion of a subtree $\mathcal{S}$, respectively. The objective is to find the optimum subtree $\mathcal{S}_o$ that has the minimum distortion, such that the rate of $\mathcal{S}_o$ meets the required budget constraint. In other words, the optimum subtree $\mathcal{S}_o$ must satisfy the following:

$$\mathcal{D}(\mathcal{S}_o) = \min_{\mathcal{T} \Rightarrow \mathcal{S}} \mathcal{D}(\mathcal{S}) \tag{3.1}$$

such that the following budget constraint is met:

$$\mathcal{R}(\mathcal{S}_o) \leq \mathcal{R}_{budget} \tag{3.2}$$

where $\mathcal{R}_{budget}$ is the maximum allowable bits one can use to encode the BSP tree.

The G-BFOS algorithm guarantees the detection of the optimum subtree $\mathcal{S}_o$ (with the minimum distortion), only if the tree under consideration meets a *monotonicity* constraint. Under this constraint, the distortion associated with *any* pruned tree (of the original tree) must be larger than or equal to the distortion associated with the original tree, *and* the rate of the pruned tree must be smaller than or equal to the rate of the original tree. In other words, pruning of an intermediate node should not decrease the distortion *and* should not increase the rate.

The monotonicity constraint, however, does not always hold for tree-structured data when *variable* quantizers are used for the terminating (leaf) nodes of the tree [27]. In the case of using a (nonhierarchical) uniform quantizer, we apply the pruning algorithm after populating each leaf node in the tree with the number of bits required to encode that node. In other words, we prune a BSP tree with fixed leaf-nodes' *quantizers*. Therefore, and as shown in [26], this leads to a BSP tree that meets the monotonicity constraint. However, the monotonicity constraint does not *always* hold when the hierarchical quantizer (explained in the previous section) is used. Nevertheless, we show in the simulation results' section that using the G-BFOS pruning algorithm for BSP-tree coding (with both cases of uniform and hierarchical quantization) provides very satisfactory results in reducing the bit rate while maintaining a small increase of the encoded image-signal distortion.

Here, we provide a very brief description of the G-BFOS algorithm and how it can be used to prune BSP trees. For more details regarding the algorithm, the reader is referred to [2], [5], and [26].

In general, pruning of an intermediate node causes some changes in the rate and distortion of the tree. As mentioned above, based on the monotonicity assumption of the G-BFOS algorithm, the change-in-rate $\Delta\mathcal{R}(\mathcal{T})$ due to pruning a node t is always less than or equal to zero: $\Delta\mathcal{R}(\mathcal{T}) \leq 0$. Similarly, the change-in-distortion, due to the pruning of t, is always larger than or equal to zero:$\Delta\mathcal{D}(\mathcal{T}) \geq 0$. An important parameter to the G-BFOS algorithm is the ratio of the increase-in-distortion to the decrease-in-rate, i.e.,

$$\lambda(t) = -\frac{\Delta\mathcal{D}(\mathcal{T})}{\Delta\mathcal{R}(\mathcal{T})}. \tag{3.3}$$

From this definition, one can see that $\lambda(t)$ is a positive number. In addition, and for an obvious reason, the parameter λ is commonly referred to as the *slope magnitude* of the rate-distortion function, or simply the *slope* parameter. Since the objective is to maximize the reduction in the bit rate (i.e., minimize $1/\Delta\mathcal{R}(\mathcal{T})$), while minimizing the increase in the distortion (i.e., minimizing $\Delta\mathcal{D}(\mathcal{T})$), it is natural that one should minimize the slope parameter $\lambda(t)$. And that is exactly what the G-BFOS algorithm accomplishes, taking advantage of the monotonicity condition explained above. It is important to note that for every (nonleaf) node t in the tree, one can compute a corresponding parameter $\lambda(t)$ that measures the overall impact that the pruning of the node t has on the total rate and distortion of the whole tree.

In summary, at every step of the G-BFOS algorithm, one has to find the node t with the minimum slope parameter $\lambda(t)$. After detecting the desired node (with the minimum slope), that node is pruned from the tree. This pruning process, which generates a new subtree at every step, is continued until the total rate of the tree meets the desired budget constraint (i.e.,$\mathcal{R}(\mathcal{S}) \leq \mathcal{R}_{budget}$).

IV. BSP-Tree Compression Results and Comparison

The BSP-tree compression techniques explained in this paper were simulated in software, and applied to several images. In the following, we compare our BSP-tree-based compression method with other coding schemes. Here, we report the performance numbers of both nonsegmentation- and segmentation-based methods. For this comparison between our work and other compression techniques, we use the 512×512 Lenna image, which is the most common test image used in the compression literature. In addition, this image represents a human face (for which distortion is very noticeable), and includes different types of textures.

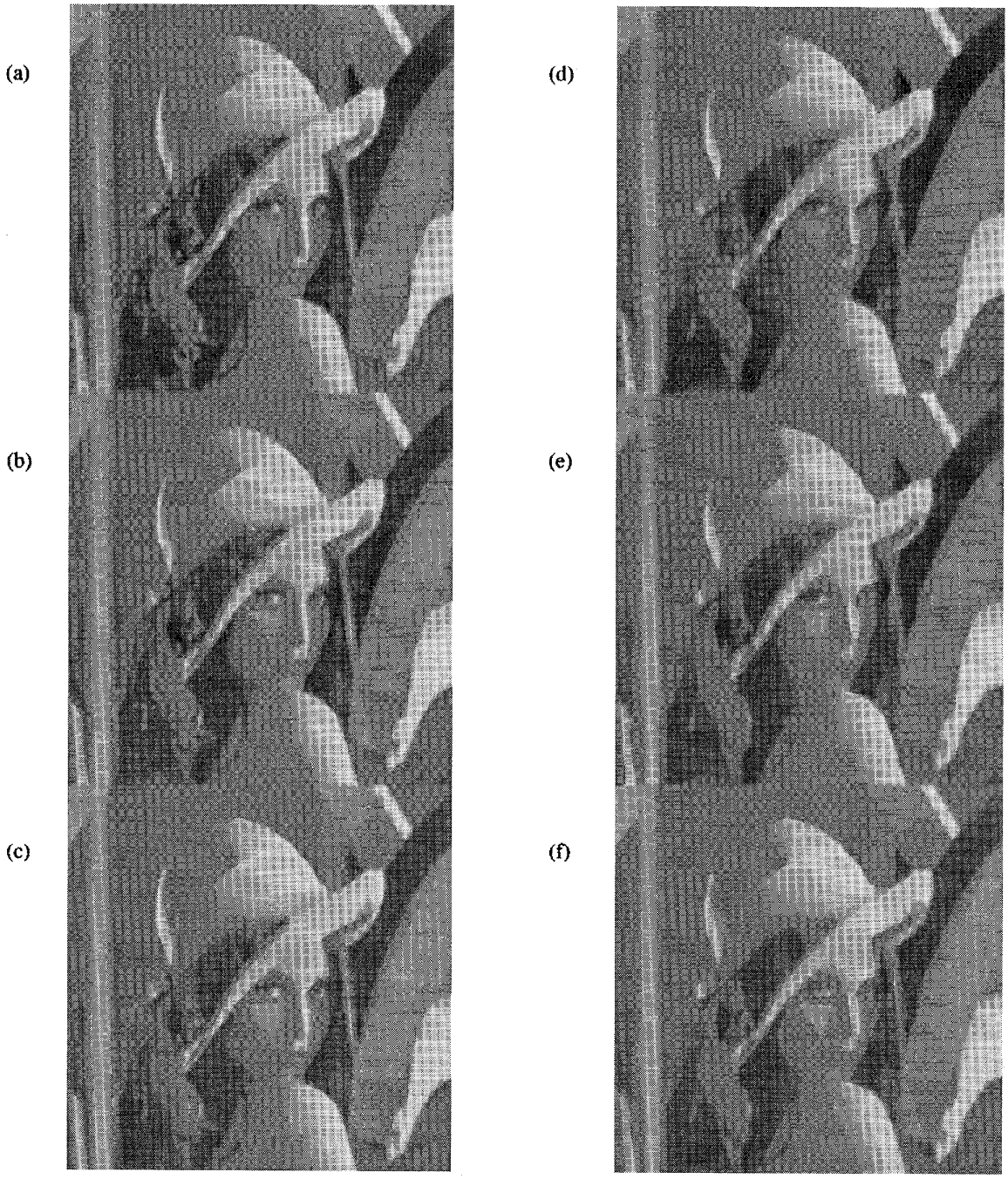


Fig. 11. Several decoded images of Lenna using different uniform quantizers and target bit rates. Images (a)–(c) are coded with 0.1 b/pixel, a quantizer range of 64 for the first-order polynomial slopes and, and the following number of levels: (a) 16, (b) 64, and (c) 512. The images in (d)–(f) are coded with a 0.05 b/pixel. The images in (d) and (e) are coded with a range value of 4 and number of levels of 16 and 512, respectively. The image in (f) is coded using the mean values of the pixel intensities within each cell (i.e., zero-order approximation).

Fig. 11 shows several decoded images of Lenna using the BSP-tree compression method described in this paper. The original BSP-tree representation was generated using the LSE-based algorithm described in [25]. Fig. 11(a)–(c) are all coded with a bit rate of 0.1 b/pixel. Fig. 11(d)–(f) are coded with a bit rate of 0.05. It is clear from the figure that changing the parameters of the uniform quantizer can have a significant impact on the quality of the final decoded images. These images were generated from a high-rate (around 0.2 b/pixel) image using the G-BFOS pruning algorithm.

Fig. 12 shows both BSP-tree images coded with the adaptive quantizer, and Joint Photographers Expert Group (JPEG) images coded at the same data rates of 0.07 and 0.1 b/pixel. It is clear that, at these low rates, the BSP-tree approach

Fig. 12. Comparison of BSP-tree and JPEG images. (a) and (b) represent BSP tree images coded at 0.1 ($QF = 8$) and 0.7 ($QF = 32$) b/pixel, and with PSNR values of 25.7 and 24.7 dB, respectively, using adaptive quantization for the first-order approximation of the cell attributes. (c) and (d) are JPEG images coded with bit rates of 0.1 and 0.07 b/pixel and PSNR values of 25.6 and 24.1 dB, respectively.

provides superior results when compared with JPEG. Also, by comparing the BSP images in Figs. 11 and 12, one can observe an improvement in the image quality when using the adaptive (hierarchical) quantization approach.

We use the PSNR as a measure of quality when comparing the different coding methods, as follows:[13]

$$\text{PSNR} = 10 \log \frac{V^2}{\text{MSE}} \tag{4.1}$$

[13] As mentioned earlier, we do not believe that the PSNR measure is a good criterion for comparison. However, it is the only one used consistently in the compression literature.

when V is the peak-to-peak value of the pixel intensities within the test image. For images with $M \times N$ pixels, MSE is the mean square of the difference (error) signal between the original image $I(x, y)$and the decoded image $\hat{I}(x, y)$, as follows:

$$\text{MSE} = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [I(x, y) - \hat{I}(x, y)]^2. \tag{4.2}$$

Since we are testing a gray-level image with an 8-b/pixel representation, we use $V = 255$. The PSNR of the decoded Lenna images shown in Fig. 11 (with uniform quantizers)

ranges from 23.3 dB (for 0.05 b/pixel) to 25 dB (for 0.1 b/pixel).

Under the traditional (nonsegmentation) works, in addition to JPEG, we looked at the performances of subband and vector quantization-based methods. For some of the JPEG algorithm performance (for 0.25 b/pixel and above), we have used the results shown in Table 10.3 in [20]. The JPEG performance numbers range from a PSNR of around 30.81 dB at 0.25 b/pixel to around 39.95 dB at 1.5 b/pixel.

One of the best subband coding results for the Lenna image is given by [3], which shows a PSNR of 31 dB at 0.27 b/pixel. In [20] the authors report the results of using a hybrid subband/vector-quantization approach. A range of 30–35 dB PSNR are given for 0.38–0.94 b/pixel compression rates. Better performance numbers are reported in [10] for a hybrid SB–VQ method with a PSNR of 30 dB at 0.25 bpp.

Regarding VQ coding schemes, we considered the variable-rate multistage hierarchical VQ (MSHVQ) approach by Ho and Gersho [6], and the full-search VQ method with pruned tree structured quantizers [28]. For the Lenna image, the MSHVQ, and full-search VQ methods reported PSNR's of 30.93 at 0.37 b/pixel and 29.29 dB at 0.32 b/pixel, respectively.

We have also looked at the performance of more recent nonsegmentation-based coding schemes such as the wavelet [1] and fractal-block [8] image compression methods. In [1], PSNR's of 30.85 dB at 0.37 b/pixel and 29.11 at 0.21 b/pixel are reported. In [8], the author divides the image into small square blocks (e.g., 8×8) and then apply fractal coding (iterated transformations) to these blocks. The author reports a PSNR of 31.4 dB at 0.6 b/pixel.

As discussed before, the segmentation-based coding scheme that is most related to our BSP-tree method is the ATSS approach [34]. To compare the compression performance of both the ATSS and BSP-tree approaches, one must look at the same data (Lenna image) using the same error criterion. Although simulation results for Lenna were reported in [34], the author did not provide any error performance numbers for the decoded images. However, based on a subjective evaluation of the decoded images shown in [34], one can observe that the BSP-tree approach provides better compression performance for the same bit rate. (Compare Fig. 8 in [34] with the decoded Lenna images shown in here.) Or, one can observe a lower bit rate for our approach for the same quality images.[14] (Compare Fig. 8(a) and (b) in [34] with the BSP-tree images in Figs. 11 and 12.)

The other segmentation-based approach that we considered (for the sake of comparison) is the quadtree-based coding scheme. Many people have employed quadtrees for image coding applications [14] and [31]. Here, we focused on the work by Vaisey and Gersho, since it represents the latest research in that area, and it also provides the best quadtree-based compression results when compared to the other works. In their approach, Vaisey and Gersho segment the image into variable-size blocks, and classify each block into perceptually important region (small blocks) or random texture regions (large blocks). Under their variable block size segmentation (VBSS) algorithm, mean and residual images are generated and coded separately, where both images are segmented using the quadtree approach. The residual image is classified into low detail, high detail, and random texture regions. The small high-detail regions are coded using classified VQ, whereas the large low-detail regions are coded with a hybrid transform coding/VQ scheme. Under this approach, the authors reported bit rates between 0.277 b/pixel with 30.20 dB PSNR and 0.36 b/pixel with 31.36 PSNR.

Fig. 13 shows the PSNR performance numbers for some of the compression methods outlined above. All data points shown in the figure are for the 512×512 Lenna image. The following observations can be made from the figure:

- In general, for high-quality (around and more than 30 dB), high-rate (around and higher than 0.25 b/pixel) compression applications, the mainstream coding schemes (e.g., SB, VQ, DCT) provide better solution than segmentation-based methods (including the BSP-tree approach). In other words, we believe that the computational complexity and the PSNR performance numbers reported in the literature for the different segmentation-based methods do not justify using these approaches.
- The recent quadtree-based approach by Vaisey and Gersho gives very good results when compared with other quadtree and segmentation based methods. The encoding of a residual image and the classification of the different regions within that image have contributed in providing good PSNR performance results. These ideas can be extended to our approach (i.e., encoding of a BSP-tree residual image and classifying the image attributes within the resulting cells). We believe that these enhancements can improve our performance numbers as well.
- For low bit-rate coding applications, we believe that the BSP-tree approach provides very promising results. In addition, this approach provides a clear advantage over other low and high bit-rate coding schemes for applications requiring geometric transformations. As explained in detail in [26], due to the flexibility of the BSP tree representation, all line-preserving transformations can be applied to the encoded image.

V. Conclusion

In this paper, we have described a BSP-tree-based image coding system that is capable of achieving high compression ratios when applied on still images. The coding method presented here is based on a hierarchical approach for encoding the partitioning lines of the BSP-tree representation of images. We also described how to efficiently encode the BSP-tree cells using low-order polynomials for approximating the image signal within these cells. Moreover, an optimum pruning algorithm was used to reduce the bit rate of the encoded BSP

[14] It is important to note that the computational expense of the BSP-tree approach presented here is higher than the ATSS method. As discussed in [34], the computational complexity of the ATSS approach is on the order of N^2 for an $N \times N$ image. For the same image, the LSE-based BSP-tree construction method has a computational complexity on the order of N^3 [25]. However, the computational expense of the BSP-tree approach can be reduced (by about an order of magnitude) when employing a multiresolution method [24]. The use of a multiresolution approach for an LSE-based BSP-tree image compression requires further study.

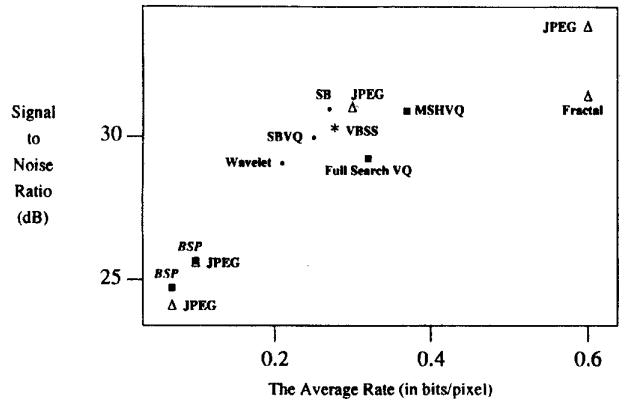


Fig. 13. PSNR performance comparison of the BSP-tree coding method with other compression schemes. Please note that although the PSNR values of the BSP-tree results are close to the JPEG PSNR numbers, the JPEG images are significantly worse than the corresponding BSP-tree images (as shown in Fig. 12).

tree while minimizing distortion. As shown in the simulation results section of this paper, the overall BSP-tree-based coding approach provides very high compression ratios. For example, it is shown that one can encode a gray-level, complex image (of a human face) using less than 0.1 b/pixel, which represents a compression ratio of more than 80.

REFERENCES

[1] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. Image Processing,* vol. 1, no. 2, pp. 205–220, Apr. 1992.

[2] P. A. Chou, T. Lookabaugh, and R. M. Gray, "Optimal pruning with applications to tree-structured source coding and modeling," *IEEE Trans. Inform. Theory,* vol. 35, pp. 299–315, Mar. 1989.

[3] J. Darraph and R. Baker, "Fixed distortion, variable rate subband coding of images," in *Proc. SPIE Int. Soc. Opt. Eng.,* vol. 1001, Dec. 1988, pp. 979–990.

[4] R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," *CACM,* vol. 15, pp. 11–15, 1972.

[5] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression.* Norwell, MA: Kluwer, 1992.

[6] Y. Ho and A. Gersho, "Variable-rate multistage vector quantization for image coding," in *ICASSP Proc.,* 1988, pp. 1156–1159.

[7] J. Illingworth and J. Kittler, "A survey of the Hough transform," *Comput. Vision, Graphics, Image Processing,* vol. 44, pp. 87–116, 1988.

[8] A. Jacquin, "Fractal image coding: A review," *Proc. IEEE,* vol. 81, no. 10, pp. 1451–1465, Oct. 1993.

[9] A. K. Jain, "Image data compression: A review," *Proc. IEEE,* vol. 69, pp. 349–389, Mar. 1981.

[10] C. S. Kim *et. al.,* "An improved SBC/VQ scheme for color image coding," in *ICASSP Proc.,* 1989, pp. 1941–1944.

[11] M. Kocher and M. Kunt, "A contour-texture approach to image coding," in *ICASSP Proc.,* 1982, pp. 436–440.

[12] M. Kunt, A. Ikonomopoulos, and M. Kocher, "Second generation image coding techniques," *Proc. IEEE,* vol. 73, pp. 549–574, Apr. 1985.

[13] M. Kunt, M. Benard, and R. Leonardi, "Recent results in high-compression image coding," *IEEE Trans. Circuits Syst.,* vol. CAS-34, no. 11, pp. 1306–1336, Nov. 1987.

[14] R. Leonardi and M. Kunt, "Adaptive split-and-merge for image analysis and coding," *Proc. SPIE,* vol. 594, 1985.

[15] S. G. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Trans. Pattern Anal. Machine. Intell.,* vol. 11, pp. 674–693, July 1989.

[16] H. G. Musmann, P. Pirsch, and H. Grallet, "Advances in picture coding," *Proc. IEEE,* vol. 73, pp. 523–548, Apr. 1985.

[17] A. N. Netravali and J. O. Limb, "Picture coding: A review," *Proc. IEEE,* vol. 68, pp. 366–406, Mar. 1980.

[18] A. N. Netravali and B. G. Haskell, *Digital Pictures: Representation and Compression.* New York: Plenum, 1988.

[19] W. K. Pratt, *Digital Image Processing.* New York: Wiley, 1978.

[20] M. Rabbani and P. W. Jones, *Digital Image Compression Techniques.* Bellingham, WA: SPIE Press, 1991.

[21] H. Radha, "Least-square binary partitioning of L_2 functions with convex domains," AT&T Bell Labs. Tech. Memo, Draft, Sept. 1990.

[22] H. Radha, R. Leonardi, B. Naylor, and M. Vetterli, "Image representation using binary space partitioning trees," in *Proc. Visual Commun. Image Processing V,*vol. 1360, pt. 1, Oct. 1990, pp. 639–650.

[23] H. Radha, R. Leonardi, and M. Vetterli, "A multi resolution approach to binary tree representations of images," in *ICASSP Proc.,* May 1991, pp. 2653–2656.

[24] H. Radha, R. Leonardi, M. Vetterli, and B. Naylor, "Binary space partitioning (BSP) tree representation of images," *J. Vis. Commun. Image Repres.,* pp. 201–221, Sept. 1991.

[25] H. Radha, M. Vetterli, and R. Leonardi, "Fast piecewise constant approximation of images" in *Vis. Comm. Image Proc. 91: Vis. Comm.,* Nov. 1991, vol. 1605, pt. 1, pp. 475–486.

[26] H. Radha, "Efficient image representation using binary space partitioning trees," Ph.D. dissertation, Department Electr. Eng., Columbia Univ., NY, 1993.

[27] K. Ramchadran and M. Vetterli, "Best wavelet packet bases in a rate-distortion sense," *IEEE Trans. Image Processing,* pp. 160–175, Apr. 1993.

[28] E. A. Riskin and R. M. Gray, "A greedy tree growing algorithm for the design of variable rate vector quantizers," in *Proc. Picture Coding Symp.,* Cambridge, MA, 1990.

[29] W. F. Schreiber, C. F. Knapp, and N. D. Kay, "Synthetic highs: An experimental TV bandwidth reduction system," *J. Soc. Motion Picture TV Eng.,* vol. 68, pp. 525–537, 1959.

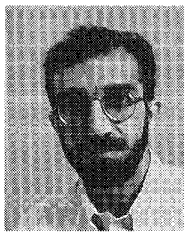
[30] G. J. Sullivan and R. L. Baker, "Efficient quadtree coding of images and video," in *ICASSP Proc.,* May 1991, pp. 2661–2664.

[31] J. Vaisey and A. Gersho, "Image compression with variable block size segmentation," *IEEE Trans. Signal Processing,* vol. SP-40, pp. 2040–2060, Aug. 1992.

[32] M. Vetterli and K. M. Uz, "Multiresolution coding techniques for digital video: A review," in *Special Issue Multidimen. Processing Video Signals, Multidimen. Syst. Signal Processing.* Norwell, MA: Kluwer, 1992, pp. 161–187.

[33] J. Woods and S. O'Neil, "Subband coding of images," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-34, pp. 1278–1288, Oct. 1986.

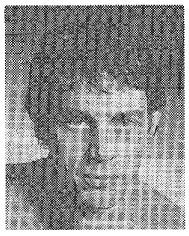
[34] X. Wu, "Image coding by adaptive tree-structured segmentation," *IEEE Trans. Inform. Theory,* vol. 38, pp. 1755–1767, Nov. 1992.



Hayder Radha (S'85–M'86–S'87–M'92) received the B.S. with Honors in electrical engineering from Michigan State University in 1984, the M.S. from Purdue University's School of Electrical Engineering, West Lafayette, IN, in 1986, and the Ph.D. in electrical engineering from the Center for Telecommunications Research and Department of Electrical Engineering at Columbia University, New York, NY in 1993.

He is a Principal Member of Research Staff at Philips Research Laboratories, Briarcliff Manor, NY. Prior to joining Philips Research, he was a Distinguished Member of Technical Staff at AT&T Bell Laboratories. He started his career with Bell Labs, Holmdel, NJ, in 1986, and while at Bell Labs (1986 to 1996), he worked in the areas of broadband audiovisual communications systems, parallel architectures, and algorithms for image processing and computer graphics applications, and video compression and processing for video-telephony and teleconferencing services. During the academic year of 1985 to 1986, he was a research assistant at Purdue University's School of Electrical Engineering, West Lafayette, IN. His current research interests are image representation and compression using computer vision and computer graphics techniques, video compression and processing for high-definition television, and broadband audiovisual systems.

Dr. Radha is an active member of several audiovisual standards bodies including the International Telecommunication Union (ITU) Experts Group on ATM Video Coding, and the Packetized Television Working Groups of the Society of Motion Picture and Television Engineers (SMPTE). Between 1994 to 1996, he served as the editor and cochairman of ITU-T Recommendations H.310, Broadband Audiovisual Communications Systems and Terminals, and H.321, Adaptation of Narrowband ISDN Visual Telephones to Broadband ISDN Networks.



Martin Vetterli (M'86–SM'90–F'95) received the Dipl. El.-Ing. degree from ETH Zürich (ETHZ), Switzerland, in 1981, the MS degree from Stanford University in 1982, and the Doctorat és Science degree from EPF Lausanne (EPFL), Switzerland, in 1986.

He was a Research Assistant at Stanford and EPFL, and has worked for Siemens and AT&T Bell Laboratories. In 1986, he joined Columbia University, New York, where he was last an Associate Professor of Electrical Engineering and codirector of the Image and Advanced Television Laboratory. In 1993, he joined the University of California at Berkeley where he has been a Professor in the Department of Electrical Engineering and Computer Sciences. Since 1995, he has been a Professor of Communication Systems at EPF Lausanne, Switzerland, and since 1996, he chairs the Communications Systems Division. His research interests include wavelets, multirate signal processing, computational complexity, signal processing for telecommunications, digital video processing and compression, and wireless video communications.

Dr. Vetterli is a member of SIAM and the Area Editor for Speech, Image, Video, and Signal Processing of the IEEE TRANSACTIONS ON COMMUNICATIONS. He is also on the editorial boards of SIGNAL PROCESSING, IMAGE COMMUNICATION, ANNALS OF TELECOMMUNICATIONS, APPLIED AND COMPUTATIONAL HARMONIC, ANALYSIS, and THE JOURNAL OF FOURIER ANALYSIS AND APPLICATIONS. He received the Best Paper Award of EURASIP in 1984 for his paper on multidimensional subband coding, the Research Prize of the Brown Bovery Corporation (Switzerland) in 1986 for his thesis, and the IEEE Signal Processing Society's 1991 Senior Award for a 1989 Transactions paper with D. LeGall. He was a plenary speaker at the 1992 IEEE ICASSP in San Francisco, and is coauthor, with J. Kovacevic, of the book *Wavelets and Subband Coding* (Prentice-Hall, 1995).

Riccardo Leonardi (S'80–M'87) Photograph and biography not available at the time of publication.