Rate-Distortion Optimized Tree Structured Compression Algorithms for Piecewise Polynomial Images

Rahul Shukla*†, Pier Luigi Dragotti[1], Minh N. Do‡ and Martin Vetterli†§

†Audio-Visual Communications Laboratory,
Swiss Federal Institute of Technology Lausanne, Switzerland
[1]Department of EEE, Imperial College, London, UK
‡Department of ECE, University of Illinois at Urbana-Champaign, USA
§Department of EECS, University of California Berkeley, CA 94720, USA
Email: rahul.shukla@epfl.ch, p.dragotti@imperial.ac.uk, minhdo@uiuc.edu, martin.vetterli@epfl.ch

IEEE Transactions on Image Processing

Abstract

This paper presents novel coding algorithms based on tree structured segmentation, which achieve the correct asymptotic rate-distortion (R-D) behavior for a simple class of signals, known as piecewise polynomials, by using an R-D based prune and join scheme. For the one dimensional (1-D) case, our scheme is based on binary tree segmentation of the signal. This scheme approximates the signal segments using polynomial models and utilizes an R-D optimal bit allocation strategy among the different signal segments. The scheme further encodes similar neighbors jointly to achieve the correct exponentially decaying R-D behavior $\left(D(R) \sim c_0 2^{-c_1 R}\right)$, thus improving over classic wavelet schemes. We also prove that the computational complexity of the scheme is of $O(N \log N)$. We then show the extension of this scheme to the two dimensional (2-D) case using a quadtree. This quadtree coding scheme also achieves an exponentially decaying R-D behavior, for the polygonal image model composed of a white polygon shaped object against a uniform black background, with low computational cost of $O(N \log N)$. Again, the key is an R-D optimized prune and join strategy. Finally, we conclude with numerical results, which show that the proposed quadtree coding scheme outperforms JPEG2000 by about 1 dB for real images, like cameraman, at low rates of around 0.15 bpp.

*Corresponding author. Address: see above; Phone: +41 21 693 7663; Fax: +41 21 693 4312. This work was supported by the Swiss National Science Foundation under grant number 20-63664.00.

I. Introduction

The quest for improved image compression is an on-going research effort of both theoretical and practical interest. Transform coders, introduced in the 1950's [12], have played a key role, in particular with discrete cosine transform (DCT) based coding [13], [21] leading to the JPEG standard [17], and then with the wavelet transform and its inclusion into the JPEG2000 standard [28]. A good overview of transform coding is presented in [11], [26]. In the latest wavelet coders and JPEG2000, wavelets are used because of their good non-linear approximation (NLA) properties for piecewise smooth functions in one dimension [31]. However, since wavelets in 2-D are obtained by a tensor-product of one dimensional wavelets, they are adapted only to point singularities and cannot efficiently model the higher order singularities, like curvilinear singularities, which are abundant in images. This suggests that wavelets might have some limitation for image processing applications, in particular for compression.

Since geometrical features, like edges, represent one of the most important perceptual information in an image, we need new schemes capable of exploiting the geometrical information present in images. Therefore, the challenge for the image coding community is to design efficient geometrical coding schemes. From an image representation point of view, a number of new schemes have emerged that attempt to overcome the limitations of wavelets for images with edge singularities. They include, to name a few, curvelets [1], wedgelets [8], beamlets [9], contourlets [7], bandelets [18] and edge adaptive geometrical schemes [4]. Such schemes try to achieve the correct N-term NLA behavior for certain classes of 2-D functions, which can model images. To predict the performance of these schemes in image compression would require a precise R-D analysis, which is usually more difficult than NLA analysis.

Recently, there has been a growing interest in the study of piecewise polynomial functions as an approximation to piecewise smooth functions. Wavelets have long been considered ideal candidates for piecewise smooth functions due to their vanishing moment properties [16]. It was shown in [3], [19] that for piecewise polynomial signals, the squared error distortion of wavelet based coders decays as $D(R) \sim d_0\sqrt{R}2^{-d_1\sqrt{R}}$. However, since such a signal can be precisely described by a finite number of parameters, it is not difficult to observe that the R-D behavior of an oracle based method decays as

$$D(R) \sim c_0 2^{-c_1 R}. \tag{1}$$

In [19], this R-D behavior has been realized with a polynomial computational cost $\left(O\left(N^3\right)\right)$ using dynamic programming (DP). However, this scheme cannot be generalized to the 2-D case.

For image coding applications, tree segmentation based schemes have always been popular due to their low computational cost. Quadtree based image compression, which recursively divides

the image into simple geometric regions, has been one of the most popular segmentation based coding schemes investigated by researchers [15], [25], [27], [29], [33]. Leonardi *et al.* [15] utilized the classic split and merge segmentation techniques to extract image regions and then approximate the contours and image characteristics of those regions. In [14], Lee proposed adaptive rectangular tiling for image compression by using different probability models for compressing different regions of a wavelet subband. Radha *et al.* [22] presented binary space partitioning tree coding scheme, which employed parent-children pruning for searching the optimal tree structure. Recently, Wakin *et al.* [32] extended the zerotree based space frequency quantization scheme by adding a wedgelet symbol [8] to its tree pruning optimization. This enables the scheme to model the joint coherent behavior of wavelet coefficients near the edges. Another interesting work for the adaptive edge representations is reported in [30], which employs non-dyadic rectangular partitioning for image segmentation.

In the present work, our aim is to develop a computationally efficient tree based algorithm for attaining the optimal R-D behavior for certain simple classes of geometrical images, namely piecewise polynomial images with polynomial boundaries. A good approximation of this class allows to develop good approximation and compression schemes for images with strong geometrical features and, as experimental results show, also for real life images, where an improvement of about 1 dB is achieved over the state of the art image coder (JPEG2000). This shows the potential of such geometry based image coding.

The main difference between the proposed prune-join tree algorithm and the tree segmentation based schemes considered in [2], [22], [23], [27], [29], [33] is as follows: The schemes in the literature employ the parent children pruning to obtain the optimal tree structures for the given bit budget. Hence, they fail to exploit the dependency among the neighboring nodes with different parents and cannot achieve the correct R-D behavior, whereas our prune-join scheme encodes similar neighbors jointly. Thus, the prune-join coding scheme extends the concept of *pruning the children* to the *joining of similar neighbors*. In doing so, the proposed scheme achieves the optimal R-D behavior for piecewise polynomial signals. Since our algorithm achieves the optimal R-D behavior with computational ease ($O(N \log N)$), it is practical as well.

Recent work closely related to our work is the wedgelets/beamlets based schemes presented in [8], [9]. These schemes also attempt to capture the geometry of the image by using the linear-edge model explicitly in the approximation tile. The main focus of these schemes remains the efficient approximation of edges only without much attention to the efficient coding of smooth surfaces. However, our work focuses on the efficient representation of both the edges and the smooth surfaces to achieve better R-D performance. Another important difference is that the

wedgelets/beamlets based schemes utilize an NLA framework, whereas we use an R-D framework which is the correct framework for the compression problem.

The paper is organized as follows: In Section II, we study the 1-D case in detail, and show how to modify a tree based algorithm so as to achieve the optimal R-D performance for piecewise polynomial signals. Then in Section III, we show the extension of the 1-D scheme to 2-D using a quadtree based scheme. Section III also presents the R-D behavior of the proposed algorithms for a simple image model. In Section IV, we present simulation results, which show the superiority of the proposed quadtree based image coding scheme over the wavelet based coder (JPEG2000) at low bit rates. Finally, Section V offers concluding remarks.

II. 1-D scenario: Binary tree algorithms

Our goal is to implement a compression algorithm based on the modeling assumption that signals are piecewise smooth functions. In this case, if we segment the signal into smaller pieces, then each piece can be well represented by a simpler signal model, which we choose to be a polynomial function.

In the next subsection, we consider the pruned binary tree decomposition of the signal, where two children nodes can be pruned to improve R-D performance. Then, we propose an extension of this algorithm which allows the joint-coding of similar neighboring nodes. To highlight the intuitions and the main ideas of these algorithms, we present them together with a toy example (i.e., compression of a piecewise linear signal with one discontinuity).

In Sections II-C and II-D, we formally compute the R-D performance of these two coding schemes. Section II-E presents their computational complexity. Most importantly, we show that the prune-join tree algorithm, which jointly encodes the similar neighbors, achieves optimal R-D performance (Theorem 2, Section II-D) with computational ease (Section II-E).

A. Binary tree algorithms

Consider the simple signal shown in Figure 1. It represents a piecewise linear signal with only one discontinuity at t_0. This signal has a finite number of degrees of freedom, since it is uniquely determined by the two polynomials and the discontinuity location. Assume that an oracle provides us the polynomial coefficients and the discontinuity location. Then, a compression algorithm that simply scalar quantizes these parameters achieves an exponentially decaying R-D behavior ($c_0 2^{-c_1 R}$) at high rates. In general, for signals with finite number of parameters, an oracle based method will provide an exponentially decaying R-D behavior at high rates. We will describe the oracle method in more detail in Section II-B.

Our target is to develop a compression algorithm based on the binary tree decomposition

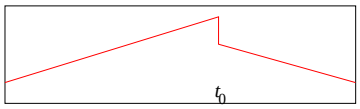


Fig. 1. A piecewise linear signal with only one discontinuity.

which achieves the oracle like R-D performance for piecewise polynomial signals (PPSs). We first consider the prune binary tree algorithm. This algorithm is similar in spirit to the algorithm proposed in [23] for searching the best wavelet packet bases. In our algorithm, each node of the tree is coded independently and, as anticipated before, each node approximates its signal segment with a polynomial. Finally the prune tree algorithm utilizes rate-distortion framework with an MSE distortion metric. This algorithm can be described as follows:

Algorithm 1: **The prune binary tree coding algorithm**

Step 1: Initialization

1. Segmentation of the input signal using the binary tree decomposition up to a tree depth $\widehat{J}$.[1]

2. Approximation of each node by a polynomial $p(t)$ of degree $\leq P$ in the least square error sense.

3. Generation of the R-D curve for each node by approximating the node by the quantized polynomial $\widehat{p}\,(t)$, which is obtained by scalar quantizing the polynomial coefficients.[2]

Step 2: The Lagrangian cost based pruning

4. For the given operating slope $-\lambda$, R-D optimal pruning criterion is as follows: Prune the children if the sum of the Lagrangian costs of the children is greater than or equal to the Lagrangian cost of the parent. That means the children are pruned if $(D_{C_1} + D_{C_2}) + \lambda(R_{C_1} + R_{C_2}) \geq (D_p + \lambda R_p)$. This criterion is used recursively to do fast pruning from the full tree depth towards the root to find the optimal subtree for a given λ [23]. The Lagrangian cost based pruning method is illustrated in Figure 2.

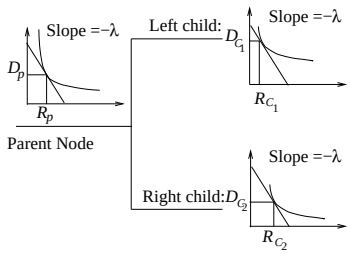


Fig. 2. Lagrangian cost based pruning criterion for an operating slope $-\lambda$ for each parent node of the tree: Prune the children if $(D_{C_1} + D_{C_2}) + \lambda(R_{C_1} + R_{C_2}) \geq (D_p + \lambda R_p)$.

[1]In the paper, we use J to indicate the final tree-depth for a given bit-budget, whereas $\widehat{J}$ indicates the initial chosen depth. Clearly, $J \leq \widehat{J}$.

[2]This is best done in an orthogonal basis, that is, the Legendre polynomial basis. We will explain this in detail in Sections II-E and IV-A.

5. Each leaf of the pruned subtree for a given λ has an optimal rate choice and the corresponding distortion. Summing up the rates of all the tree leaves along with the tree segmentation cost will provide the overall bit-rate $R^*(\lambda)$. Similarly, summing up the associated distortions of all the tree leaves will give the net distortion $D^*(\lambda)$.

Step 3: Search for the desired R-D operating slope

The value for λ is determined iteratively until the bit-rate constraint R_0 is met as closely as possible. The search algorithm exploits the convexity of the solution set and proceeds as follows [23]:

6. First determine $\lambda_{\min}$ and $\lambda_{\max}$ so that $R^*(\lambda_{\max}) \leq R_0 \leq R^*(\lambda_{\min})$.

If the inequality above is an equality for either absolute slope value, then stop. We have an exact solution, otherwise proceed to the next line.

7. $\lambda_{\text{new}} = (D^*(\lambda_{\min}) - D^*(\lambda_{\max}))/(R^*(\lambda_{\max}) - R^*(\lambda_{\min}))$.
8. Run the Lagrangian cost based pruning algorithm (Step 2) for λ_{new}.
   if $(R_0 = R^*(\lambda_{\text{new}}))$, then the optimum is found. Stop.
   elseif $(R_0 < R^*(\lambda_{\text{new}}))$, then $\lambda_{\min} = \lambda_{\text{new}}$ and go to the line 7.
   else $\lambda_{\max} = \lambda_{\text{new}}$ and go to the line 7.

The pruned binary tree decomposition of the piecewise linear function, shown in Figure 1, is depicted in Figure 3. One can observe that the prune tree scheme could not merge the neighboring nodes representing the same information (e.g., nodes $(2,3)$ and $(3,5)$), as they belong to different parents. Since this coding scheme fails to exploit the dependency among neighbors in the pruned tree, it is bound to be suboptimal and cannot achieve the oracle R-D performance.

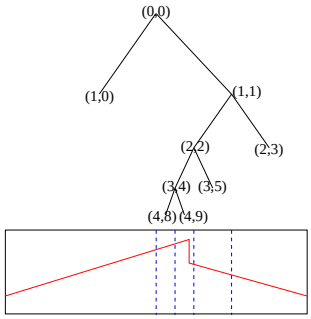


Fig. 3. The pruned binary tree segmentation.

For correcting the suboptimal behavior, we propose a prune-join coding scheme, which exploits the dependency among neighboring leaves even if they belong to different parents. This scheme extends the concept of *pruning the children* to the *joining (merging) of similar neighbors*.

This new scheme employs the prune tree coding scheme followed by the neighbor joint coding algorithm, which can be described as follows: Given the pruned tree obtained from Algorithm 1,

the neighbor joint coding is performed on the leaves of the tree. Suppose that n_j^i (or (j,i)) represents the i^{th} node at the j^{th} level of the binary tree. The pruned tree is scanned from left to right and top to bottom. For instance, the leaves of the tree shown in Figure 3 will be scanned in the following order: $(1,0),(2,3),(3,5),(4,8),(4,9)$. Assume that the current leaf is n_j^i, then the indices (i_0) of the neighbors $\left(n_{j_0}^{i_0}\right)$ at level j_0 can be computed as follows:

$$\begin{aligned} \text{Left neighbor}: \ i_0 &= 2^{(j_0-j)}i - 1; \\ \text{Right neighbor}: \ i_0 &= 2^{(j_0-j)}\,(i+1)\,; \end{aligned}$$

In the above formulation, n_0^0 is assumed to be the root node.

For R-D optimality, all leaves of the tree must operate at a constant slope point $-\lambda$ on their R-D curves. Therefore, if the algorithm finds an already scanned neighboring leaf, then it will decide about the joining of the leaves using the following Lagrangian cost based approach: The two neighbors (call them n_1 and n_2) will be joined if the sum of the Lagrangian costs of the neighbors is greater than or equal to the Lagrangian cost of the joint block (n_{Joint}), i.e., if $(D_{n_1} + \lambda R_{n_1}) + (D_{n_2} + \lambda R_{n_2}) \geq D_{n_{\text{Joint}}} + \lambda R_{n_{\text{Joint}}}$. If neighbors are jointly coded, then the neighbor joint coding variable will be set to one and the joint leaf polynomial information is stored in place of the neighbors, otherwise the neighbor joint coding variable will be set to zero and the leaf information will be stored. Note that once a joint block is constructed, it will be treated as a leaf in place of its constituent leaves for further joining operation. If the algorithm does not find any scanned neighbor, then the leaf information will be stored.

Now, if the current leaf is not the last leaf of the pruned tree, then the algorithm will restart the above described neighbor search and join operation for the next leaf of the pruned tree. Clearly, the neighbor joint coding variable is an indicator functional, which keeps track of the neighbor joining information of the pruned tree leaves. Thus, each leaf has a binary neighbor joint coding variable, which indicates whether it is jointly coded or not. The prune-join coding scheme can be summarized as follows:

Algorithm 2: **The prune-join binary tree coding algorithm**

Step 1: Initialization

Following Steps 1 and 2 of Algorithm 1, find the best pruned tree for a given λ.

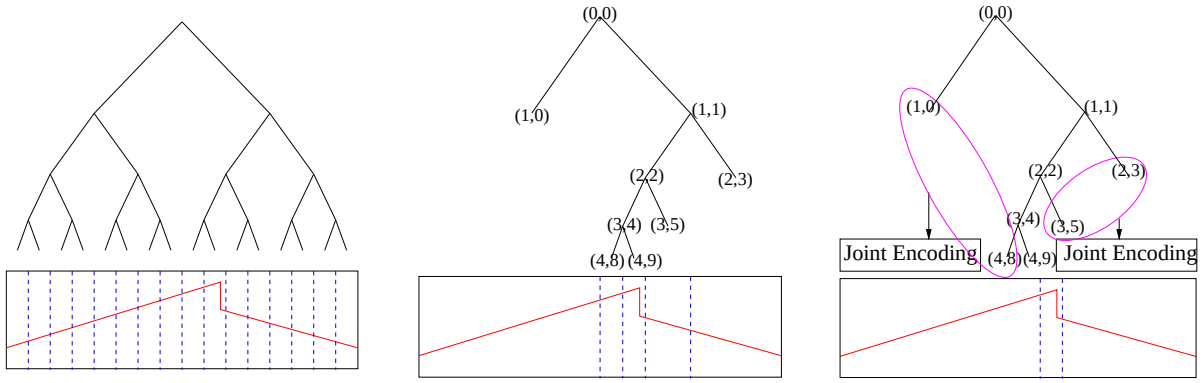
Step 2: The neighbor joint coding algorithm

Given the pruned tree, perform the joint coding of similar neighboring leaves as explained above.

Step 3: Search for the desired R-D operating slope

Similar to Algorithm 1, iterate the process over λ until the bit budget constraint is met.

It is clearly visible in Figure 4(c) that the prune-join coding scheme is essentially coding a



(a) The full binary tree. (b) The prune binary tree. (c) The prune-join binary tree.

Fig. 4. Comparative study of different tree segmentation algorithms.

fixed number of blocks like the oracle method. Therefore, we expect it to achieve the oracle like R-D performance for piecewise polynomial signals.[3]

B. R-D analysis of the oracle method

Consider a continuous time piecewise polynomial signal $f(t)$, defined over the interval $[0, T]$, which contains S internal singularities. Assume that the function $f(t)$ is bounded in magnitude by some constant A and the maximum degree of a polynomial piece is P. The signal is uniquely determined by $(S+1)$ polynomials and by S internal singularities. That means such a signal can be precisely described by a finite number of parameters. Suppose that the values for the parameters of the polynomial pieces, and the locations of the internal singularities are provided with arbitrary accuracy by an oracle. In that case, it has been shown in [19] that the R-D behavior of the oracle based method decays as

$$D(R) \leq c_0 2^{-c_1 R}, \tag{2}$$

where $c_0 = 2A^2T(S+1)(P+1)^2$ and $c_1 = \frac{2}{(P+3)(S+1)}$.

C. R-D analysis of the prune binary tree coding algorithm

This section presents the asymptotic R-D behavior of the prune binary tree coding algorithm for piecewise polynomial signals. We compute the worst case R-D upper-bound in the operational (algorithmic) sense. First, we show that this algorithm results in a number of leaves to be coded which grows linearly with respect to the decomposition depth J. This implies that several nodes

[3]However, note that this scheme may not find the globally optimal solution to the joint coding problem. The reason is that the pruning step may decide to keep a node because the cost of coding its children is higher, whereas in fact this cost may be much lower than expected due to the neighbor joint coding scheme which operates later.

with same parameters are coded separately (e.g., see Figure 4(b)). Then, we prove that this independent coding of similar leaves results in a suboptimal R-D behavior given by Theorem 1.

Lemma 1: The bottom-up R-D optimal pruning method results in a binary tree with the number of leaves upper-bounded by $(J+1)\,S$, where J and S represent the final tree-depth and the number of internal singularities in the piecewise polynomial signal, respectively.

Proof: Since we are interested in the asymptotic R-D behavior, we will consider the worst case scenario. As the signal has only S transition points, at most S tree nodes at a tree level will have a transition point and the remaining nodes will be simply represented by a polynomial piece without any discontinuity. Clearly, at high rates, for achieving better R-D performance the tree pruning scheme will only split the nodes with singular points, as they cannot be well approximated by a polynomial.[4] This means that every level, except the levels $j=0$ and J, will generate at most S leaves. The level J will have $2S$ leaves, while the level 0 cannot have any leaf at high rates for $S>0$. Hence, the total number N_0 of leaves in the pruned binary tree is

$$N_0 \;\leq\; 2S + (J-1)\,S = (J+1)\,S. \tag{3}$$

Therefore, the number of leaves to be coded grows linearly with respect to the depth J. $\diamond$

Moreover, it can also be noted that in the pruned tree, every tree level can have at most $2S$ nodes. Hence, the total number M_0 of nodes in the pruned tree can be given as follows

$$M_0 \;\leq\; 2JS + 1. \tag{4}$$

Theorem 1: The prune binary tree coding algorithm, which employs the bottom-up R-D optimization using the parent-children pruning, achieves the following asymptotic R-D behavior

$$D_P\,(R) \leq c_2 \sqrt{R}\, 2^{-c_3\sqrt{R}}, \tag{5}$$

where $c_2 = 16A^2TS\,(P+1)^2\,\sqrt{\frac{4}{(P+1)S}}$ and $c_3 = \sqrt{\frac{4}{(P+1)S}}$, for piecewise polynomials signals.

Proof: Since the piecewise polynomial function $f\,(t)$ has only S transition points, at most S leaves will have a transition point and the remaining JS leaves (Lemma 1) can be simply represented by a polynomial piece without any discontinuity. At high rates, leaves with singular points will be at the tree depth J, so the size of each of them will be $T2^{-J}$. The distortion of each of these leaves can be bounded by $A^2T2^{-J} \leq A^2T(P+1)^2 2^{-J}$ and it will not decrease with the rate. This is because simple polynomials cannot represent piecewise polynomial functions. Leaves without singularities can be well approximated by a polynomial. In particular, a leaf l at tree level j is of size $T_l = T2^{-j}$ and its R-D function can be bounded by $D_l = A^2\,(P+1)^2\,T_l 2^{-\frac{2}{P+1}R_l}$ [19].

[4]For a proof of this simple fact, refer to [24].

Since R-D optimal solution of exponentially decaying R-D functions results in equal distortion for each leaf [5], the coding algorithm will allocate same rate R_j to all the leaves without singularities at the same tree level j. As R-D optimality requires that leaves without singularities operate at a constant slope $-\lambda$ on their R-D curves, we have

$$\begin{aligned} \frac{\partial D_j}{\partial R_j} &= -\lambda, \forall j \geq 1 \\ \Rightarrow A^2 (P+1)^2 T 2^{-j} 2^{-\frac{2}{P+1}R_j} &= \frac{(P+1)\lambda}{2 \ln 2}. \end{aligned} \tag{6}$$

Equation (6) is essentially the equal distortion constraint. Let R_j and R_k be the rates allocated to the leaves without singularities at levels j and k, respectively. The equal distortion constraint for the leaves without singularities at tree levels j and k means that

$$A^2 (P+1)^2 T 2^{-j} 2^{-\frac{2}{P+1}R_j} = A^2 (P+1)^2 T 2^{-k} 2^{-\frac{2}{P+1}R_k}$$

$$\Rightarrow R_j = R_k + \frac{P+1}{2}(k-j) \tag{7}$$

$$\Rightarrow R_{J+1} = R_J - \frac{P+1}{2} < R_J, \tag{8}$$

where R_J and R_{J+1} represent the rates allocated to leaves without singularities at levels J and $J+1$, respectively. Note that the nodes with singularities will be allocated zero rate.[5]

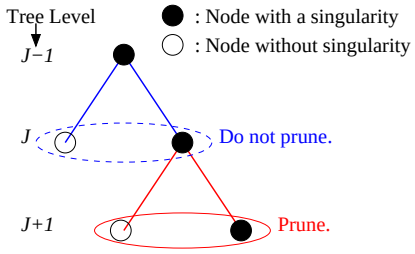


Fig. 5. Figure shows the conditions to stop the pruning of a singularity containing node at the tree level J. That means, J becomes the tree-depth.

For the given bit budget constraint, the Lagrangian cost based pruning algorithm will stop at level J if the following two conditions are satisfied (see Figure 5): (1) The Lagrangian cost of the singularity containing node at level J is less than the sum of the Lagrangian costs of its children, that is, $A^2(P+1)^2T2^{-J} < A^2(P+1)^2T2^{-(J+1)} + A^2(P+1)^2T2^{-(J+1)}2^{-\frac{2}{P+1}R_{J+1}} + \lambda R_{J+1}$, and (2) the Lagrangian cost of the singularity containing node at level $J-1$ is more than the sum of the Lagrangian costs of its children, that is, $A^2(P+1)^2T2^{-(J-1)} > A^2(P+1)^2T2^{-J} + A^2(P+1)^2T2^{-J}2^{-\frac{2}{P+1}R_J} + \lambda R_J$. These two conditions along with (6) and (8) mean that R_J must satisfy

[5]As any singularity containing node has the distortion bounded by $A^2T(P+1)^2 2^{-J}$ which will not decrease with the rate allocated to it.

the following inequality

$$\frac{1}{2} < \quad 2^{-\frac{2}{P+1}R_J}\left(1+\frac{2\ln 2}{P+1}R_J\right) \quad < 1. \tag{9}$$

This is because (6) gives $\frac{(P+1)\lambda}{2\ln 2} = A^2(P+1)^2T2^{-(J+1)}2^{-\frac{2}{P+1}R_{J+1}} = A^2(P+1)^2T2^{-J}2^{-\frac{2}{P+1}R_J}$, and (8) provides $R_{J+1} < R_J$.

Since the function $2^{-\frac{2}{P+1}R_J}\left(1+\frac{2\ln 2}{P+1}R_J\right)$ is a monotonically decreasing function of R_J for $R_J \geq 0$, we get[6]

$$\frac{P+1}{\ln 2} > \qquad R_J \qquad > 0, \text{ as } P \geq 0 \tag{10}$$

$$\Rightarrow \frac{1}{8} < 2^{-\frac{2}{\ln 2}} < \quad 2^{-\frac{2}{P+1}R_J} \quad < 1. \tag{11}$$

Multiplying the inequality (11) by $A^2T2^{-J}(P+1)^2$, we obtain

$$\frac{A^2T2^{-J}(P+1)^2}{8} < A^2T2^{-J}(P+1)^2 2^{-\frac{2}{P+1}R_J} < A^2T2^{-J}(P+1)^2. \tag{12}$$

The inequality (12) shows that the pruning scheme selects the depth J and the rate R_J such that the distortions of the leaves without singularities are of the order $O(2^{-J})$. Since the distortions of the singularity containing leaves are also of the order $O(2^{-J})$, the distortion of a leaf without singularity is comparable to that of the leaf with singularities. It is also clear from (12) that, by choosing $R_J = 0$, we will obtain the worst case R-D performance. Thus, setting $R_J = 0$ and using (7), the rate allocated to a leaf without singularity at tree level j will be given by $R_j = \frac{P+1}{2}(J-j)$. This ensures that all the leaves have a distortion of the same order $O\left(2^{-J}\right)$. Hence, the net distortion can be bounded as follows

$$\begin{aligned} D_P &\leq S\left(A^2T(P+1)^2 2^{-J}\right) + JS\left(A^2T\left(P+1\right)^2 2^{-J}\right) \\ \Rightarrow D_P &\leq A^2TS\left(P+1\right)^2\left(J+4\right)2^{-J}. \end{aligned} \tag{13}$$

Since all the tree levels, except $j = 0$, can contribute S leaves with no singularity, the total rate required for coding the leaves is

$$R_{\text{Leaves}} = S\sum_{j=1}^{J}\frac{P+1}{2}(J-j) = S(P+1)\frac{J\left(J-1\right)}{4}. \tag{14}$$

The binary tree split-merge decision variable will consume bits (R_{Tree}) equal to the total number of nodes in the pruned binary tree. Thus, (4) gives $R_{\text{Tree}} \leq 2JS + 1$. The total bit

[6]Note that substituting $R_J = \frac{P+1}{\ln 2}$ in $2^{-\frac{2}{P+1}R_J}\left(1+\frac{2\ln 2}{P+1}R_J\right)$ results in a value which is less than $\frac{1}{2}$, so we use $\frac{P+1}{\ln 2}$ to upper-bound R_J to obtain a simple analytic expression.

rate can be seen as the sum of the costs of coding the binary tree itself and the quantized model parameters of the leaves. Hence, the total bit rate can be written as follows

$$\begin{aligned} R &= R_{\text{Tree}} + R_{\text{Leaves}} \leq 2JS + 1 + \frac{(P+1)\,S}{4} J\,(J-1) \\ \Rightarrow R &\leq \frac{(P+1)\,S}{4}\,(J+4)^2\,;\ \text{as } S>0 \text{ and } J \text{ is large.} \end{aligned} \tag{15}$$

Combining (13) and (15) by eliminating J and noting that the right hand side of (13) is a decreasing function of J, whereas the right hand side of (15) is an increasing function of J, we obtain the following R-D bound

$$D_P \leq 16A^2TS\,(P+1)^2 \sqrt{\frac{4}{(P+1)\,S} R}\, 2^{-\sqrt{\frac{4}{(P+1)S}R}}.$$

Therefore, the prune binary tree algorithm exhibits the announced decay. □

Remark: The reason of the suboptimality of the prune tree algorithm is clearly visible in Lemma 1, which shows that the prune tree algorithm codes a number N_0 of leaves which grows linearly with the tree depth J. This is clearly the element in the algorithm that determines the suboptimal decay-rate $\sqrt{R}$ of the R-D function. A statistical modeling of leaves can improve the constants but cannot change the decay-rate. In fact, we have shown in [24] that the prune binary tree coding algorithm achieves an asymptotic R-D behavior which is lower bounded (in expectation) as follows

$$D_P(R) \geq c_2' \sqrt{R} 2^{-c_3'\sqrt{R}},$$

where $c_2' = \frac{8e}{3}A^2T$ and $c_3' = 2$, for piecewise polynomial signals.

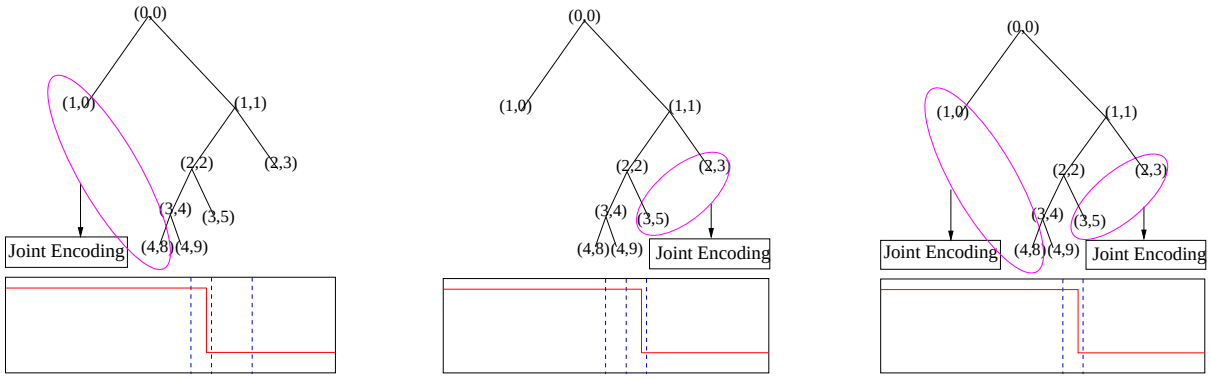
D. R-D analysis of the prune-join binary tree algorithm

Before proving that the prune-join coding scheme achieves the oracle like asymptotic R-D behavior in the operational sense, we show that this coding scheme encodes a number of leaves which remains fixed with respect to the tree depth J.

Lemma 2: The prune-join binary tree algorithm, which jointly encodes similar neighbors, reduces the effective number of leaves to be encoded to $S+1$, where S is the number of the internal singular points in the piecewise polynomial signal.

Proof: To improve the R-D performance, it is obvious that the neighbor joint coding scheme will join two neighboring leaves if the joint block does not have a singularity.[7] In particular, if J is large enough, each singularity will lie on a different dyadic leaf. Therefore, as a consequence of neighbor joining, all the leaves between any two consecutive singularity containing leaves will be joined to form a single joint block (see the example in Figure 6). Thus, the prune-join tree

[7]For a proof of this simple fact, refer to [24].



(a) Joining of left leaves. (b) Joining of right leaves. (c) Complete joining.

Fig. 6. Illustration of the prune-join binary tree joining.

algorithm results in $S+1$ joint leaves and S leaves with a singularity. Since the leaves containing a singularity will not be encoded, the number of encoded leaves becomes $S+1$. This means that the number of leaves to be coded remains constant with respect to the tree depth J. $\diamond$

Theorem 2: The prune-join binary tree algorithm, which jointly encodes similar neighbors, achieves the oracle like exponentially decaying asymptotic R-D behavior

$$D_{PJ}(R) \leq c_4 2^{-c_5 R}, \tag{16}$$

where $c_4 = 2A^2T(2S+1)(P+1)^2$ and $c_5 = \frac{2}{(S(P+7)+(P+1))}$, for piecewise polynomial signals.

Proof: The prune-join binary tree algorithm provides $(S+1)$ joint blocks and at most S leaves with a singularity. The distortion of the leaves with singularities is bounded by $A^2T2^{-J} \leq A^2T(P+1)^2 2^{-J}$ and it does not decrease with the rate (recall that the algorithm tries to approximate each block with a polynomial). The size of each joint block can be bounded by T. Thus, the distortion of each joint block is bounded by $A^2(P+1)^2 T 2^{-\frac{2}{P+1}R_l}$, where R_l is the rate allocated to that block. Again, R-D optimization forces all the joint blocks to have the same distortion. As for the prune tree algorithm, one can show that R-D optimization results in a tree-depth J and a bit allocation strategy such that the joint blocks and the singularity containing leaves have a distortion of the same order $O\left(2^{-J}\right)$. This means that the algorithm allocates $\frac{(P+1)}{2}J$ bits to each joint block and no bits to the leaves with singularities. Thus, the total rate required for coding the joint leaves is given by $R_{\text{Leaves}} = (S+1)\frac{(P+1)}{2}J$.

In the prune-join coding scheme, the side information consists of two parts: 1. Bits required to code the pruned tree (R_{Tree}). 2. Bits required to code the leaf joint coding tree ($R_{\text{LeafJointCoding}}$). The tree split-merge variable needs bits equal to the total number of nodes in the pruned tree, whereas the joint coding decision variable requires bits equal to the total number of leaves in

the pruned tree. Hence, $R_{\text{Tree}} \leq 2JS+1$ (from (4)), and $R_{\text{LeafJointCoding}} \leq (J+1)S$ (from (3)). The total bit rate is the sum of the costs of coding the binary tree itself, the leaves joint coding information and the quantized model parameters of the leaves. Thus, the total bit rate can be written as follows

$$\begin{aligned} R &= R_{\text{Tree}} + R_{\text{LeafJointCoding}} + R_{\text{Leaves}} \\ R &\leq 2JS + 1 + (J+1)S + (S+1)\frac{(P+1)}{2}J \qquad (17) \\ \Rightarrow R &\leq \frac{(S(P+7)+(P+1))}{2}(J+1). \qquad (18) \end{aligned}$$

The net distortion bound is as follows

$$\begin{aligned} D_{PJ} &\leq SA^2T(P+1)^2 2^{-J} + (S+1)A^2T(P+1)^2 2^{-J} \\ &= A^2T(2S+1)(P+1)^2 2^{-J} = c_4 2^{-(J+1)} \\ \Rightarrow D_{PJ} &\leq c_4 2^{-\frac{2}{(S(P+7)+(P+1))}R}; \text{ from (18).} \end{aligned}$$

Therefore, the prune-join tree algorithm achieves the exponentially decaying R-D behavior. □

Note that the R-D behavior of the prune-join tree scheme is worse than that of the oracle method given by (2). One can notice in (17) that the prune-join tree scheme needs $R_{\text{Tree}} = (2JS+1)$ bits to code the tree-segmentation information, which causes the divergence in the R-D performance of the proposed tree scheme and that of the oracle method.

Remark: Note that the prune tree scheme is the best in the operational R-D sense, due to the Lagrangian pruning, among all algorithms that code the dyadic segments independently. But this scheme fails to achieve the correct R-D behavior, as it cannot join the similar neighbors with different parents. On the other hand, although we cannot claim that the prune-join scheme is the best among all joint coding schemes, it achieves an exponentially decaying R-D behavior for piecewise polynomial signals as the prune-join scheme is capable of joining similar neighbors.

E. Computational complexity

For the complexity analysis, we consider a discrete time signal of size N. The complete prune tree algorithm essentially performs three operations:

1. *Initialization*: Suppose that the signal is decomposed up to the maximum tree depth $\widehat{J} = \log N$, then the number of nodes is of $O(N)$. Each tree-level ($j = 0, \ldots, \log N$) contains N pixels, which are divided among 2^j nodes. Hence, the average size of nodes is of $O(\log N)$. Initialization basically consists of the following operations:

(a) *Computation of the best Legendre polynomial approximations*: In the operational setup, for a node segment $\boldsymbol{y}$ of length L with the underlying grid $\boldsymbol{x}$, the minimum squared-error Legendre

polynomial approximation $\boldsymbol{p}$ of order P is found by solving the least square (LS) problem:

$$\min_{\boldsymbol{p}} \|V_{L,P}\,\boldsymbol{p} - \boldsymbol{y}\|^2, \tag{19}$$

(all vectors are column vectors) where $\boldsymbol{p}$ is a vector of $P+1$ polynomial coefficients and $V_{L,P}$ is the following $L \times (P+1)$ Vandermonde matrix:

$$V_{L,P} = \begin{bmatrix} \phi_0(x_1) & \phi_1(x_1) & \phi_2(x_1) & \dots & \phi_P(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \phi_2(x_2) & \dots & \phi_P(x_2) \\ & & \dots & & \\ \phi_0(x_L) & \phi_1(x_L) & \phi_2(x_L) & \dots & \phi_P(x_L) \end{bmatrix}, \tag{20}$$

where $\boldsymbol{x} = [x_1, x_2, \dots, x_L]^T$ is the underlying grid for the node and $\phi_i(x), 0 \le i \le P$, are the Legendre polynomial basis functions defined over the node-interval (x_1, x_L).[8] Note that the Legendre polynomial basis functions are computed by applying the Gram-Schmidt orthogonalization procedure on the standard polynomial basis set $\{x^0, x^1, \dots, x^P\}$. They can also be computed using Legendre polynomial recurrence relation as in [20]. We can pre-compute and store the Legendre polynomial based Vandermonde matrix $V_{L,P}$ to use for further computation. Since all the nodes of a tree level are of same size, we can assume the same underlying grid for these nodes and, thus, need to store only one Vandermonde matrix for every tree level.

The solution to the least square problem in (19) is achieved efficiently by means of a QR factorization of $V_{L,P}$ with computational cost of $O(LP)$.[9] Since the average node-size is $O(\log N)$, the overall computational cost for computing the best polynomials for all nodes will be $O(N \log N)$. Note that the polynomial degree P is included in the complexity constant.

(b) *Generation of the R-D curves*: Assume that we are utilizing R_Q different quantizers for R-D function generation. Since the computational cost of the R-D curve for a node is proportional to its size and the number of quantizers used, the overall cost of computing the R-D curves for all the tree nodes is $O(NR_Q \log N)$.

Therefore, the overall cost of computing the best polynomials and R-D curves for all the tree nodes is $O(NR_Q \log N)$.

2. *Pruning algorithm*: This requires to compute the minimum Lagrangian cost at each node for the chosen operating slope $-\lambda$. This results in a computational cost of $O(N \log R_Q)$ due to the binary search through the convex R-D curve of each node. The algorithm also performs split-merge decision at the nodes, which requires a computational cost of $O(N)$. Hence, the pruning algorithm has the computational cost of $O(N \log R_Q)$.

[8]For example, if the node-interval is $(-1, 1)$, then $\phi_0(x) = \frac{1}{\sqrt{2}}, \phi_1(x) = \sqrt{\frac{3}{2}}x, \phi_2(x) = \sqrt{\frac{45}{8}}\left(x^2 - \frac{1}{3}\right)$.

[9]QR factorization means that $V_{L,P} = QR$, with $Q \in \mathbb{R}^{L \times L}$ an orthogonal matrix and $R \in \mathbb{R}^{L \times (P+1)}$ upper triangular matrix whose last $L - P - 1$ rows are identically zero. One can find more details in [20, Chapter 3].

3. *Iterative search algorithm for an optimal operating slope*: This calls the pruning algorithm for the chosen operating slope $-\lambda$. Our bisection search scheme obtains the optimal operating slope in $O(\log N)$ iterations [23]. Thus, the computational cost of this scheme is $O(N \log R_Q \log N)$. Hence, the complete computational complexity C_{Prune} of the prune tree algorithm is

$$C_{\text{Prune}} = O(NR_Q \log N) + O(N \log R_Q \log N) \simeq O(NR_Q \log N).$$

Since a pruned binary tree has a number of leaves of $O(\log N)$ ($J \leq \log N$ and eq. (3)) and the size of any leaf is bounded by $O(N)$, the computational cost of the neighbor joint coding algorithm will be $O(NR_Q \log N)$. The prune-join coding scheme employs the prune tree algorithm followed by the neighbor joint coding algorithm. Hence, the overall computational complexity of the prune-join coding scheme is the sum of the computational costs of the prune tree scheme and the neighbor joint coding scheme. Therefore, the overall computational complexity of the prune-join coding scheme is

$$C_{\text{Prune-Join}} = O(NR_Q \log N) + O(NR_Q \log N) \simeq O(NR_Q \log N).$$

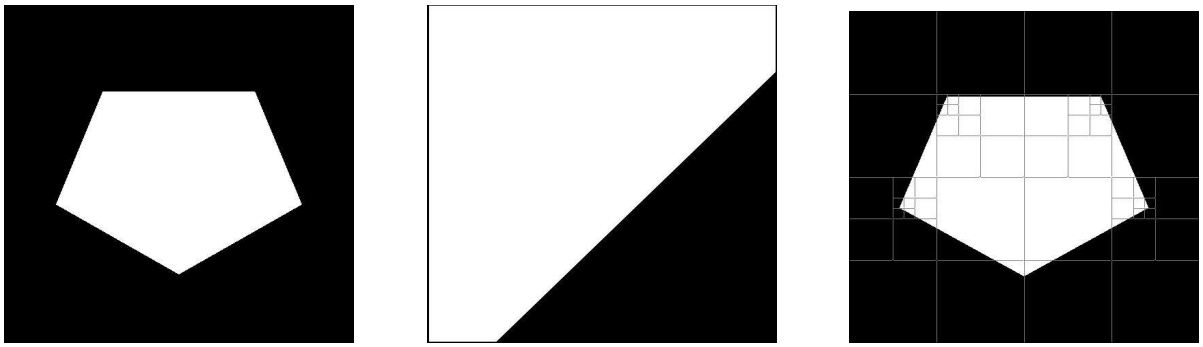
III. Extension to 2-D: Quad tree algorithms

Although the situation is much more open and complex in two dimensions, it is not hard to visualize the extension of the proposed 1-D coding scheme to the 2-D case. Clearly, all the algorithms discussed so far in 1-D have an equivalent in 2-D. The binary tree segmentation can be replaced by the quadtree segmentation and polynomial model can be replaced by the 2-D geometrical model consisting of two 2-D polynomials separated by a polynomial boundary. The Lagrangian optimization algorithm remains the same. The neighbor joint coding algorithm is more involved but it can be implemented efficiently. Therefore, we can have an efficient quadtree based coding scheme for 2-D geometrical signals.

Note that, in 1-D, the signal can contain only point-like singularities, which can be efficiently captured by the binary tree segmentation. However, in 2-D, the quadtree segmentation cannot capture the higher order edge singularities, as it can model only horizontal and vertical edges at dyadic locations. Thus, we need to improve our node-model from simple polynomial to piecewise polynomial with polynomial edge to capture the geometry inherent in the 2-D images [8].[10]

For the sake of simplicity, we carry out our analysis on a simpler image model, which we call the polygonal model. In the polygonal model, there is a white polygon shaped object against a uniform black background (see Figure 7(a)). Section III-A outlines the prune and prune-join

[10]Since the simple 2-D polynomial tile fails to capture the edge-geometry, the quadtree schemes, which use only 2-D polynomial tiles, result in a sub-optimal R-D behavior given by $D(R) \sim cR^{-1}$ for piecewise polynomial images.



(a) B/W polygonal image. (b) An edge tile. (c) Pruned quadtree.

Fig. 7. Examples of a black and white (B/W) polygonal image, an edge tile and the quadtree segmentation.

quadtree schemes. In Section III-B, we present the oracle R-D behavior. In Sections III-C and III-D, we analyze the R-D performance of the quadtree schemes. Similar to the 1-D case, we show that the prune-join quadtree scheme achieves the oracle like asymptotic R-D behavior (Theorem 4, Section III-D) with computational ease.

A. Quadtree algorithms

Similar to the 1-D case, we first consider the prune quadtree algorithm. The overall structure of this scheme is similar to the prune binary tree algorithm as described in Algorithm 1. Basically, this algorithm employs a quadtree segmentation, which approximates each node by a geometrical tile consisting of two 2-D polynomials separated by a polynomial boundary. We then perform an operational R-D optimization that is similar to the approach used for the 1-D case.

We shall describe the basic idea of the algorithm using the polygonal model. In the pruned quadtree, at each level, the only dyadic blocks that need to be divided further are the ones containing a singular point of the edge. Other dyadic blocks contain either no edge or a straight edge and they can be efficiently represented by the edge tiles shown in Figure 7(b). Essentially, the quadtree grows only in the region where the algorithm finds singular points. Thus, the quadtree recursively divides the linear edges for capturing the vertices of the polygon. Since this scheme, like the prune binary tree scheme, could not jointly code the similar nodes with different parents, it also exhibits a suboptimal R-D performance. In 2-D, there is one more ingredient for suboptimality. A vertex containing node is divided into four children, and all the children are coded separately even if two or three of them are similar. Therefore, this scheme could not perform the joint coding of similar children. This drawback can be easily seen in Figure 7(c).

For correcting the suboptimal behavior, we propose the prune-join quadtree algorithm, which performs the joint coding of similar neighboring leaves even if they have different parents. This

new scheme also allows to join two or three children only, while the prune tree scheme will either join all the children or code them independently.

The prune-join coding scheme employs the prune quadtree scheme followed by the neighbor joint coding algorithm, which decides whether neighbors should be coded jointly or independently. The neighbor joint coding scheme is similar to that of the 1-D case, except that the algorithm to search a neighbor on the quadtree is more complex. So, we shall only describe this search algorithm. Assume that the nodes n_1 and n_2 are of sizes s_1 and s_2, respectively. Suppose

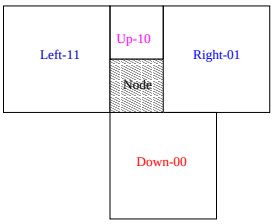


Fig. 8. 4-connected neighboring nodes. Every neighbor is assigned a two bit index.

that their origins (bottom-left points) are (x_1, y_1) and (x_2, y_2), respectively. Figure 8 shows the 4-connected neighbors of a node. The following pseudo code determines whether n_2 is a neighbor of n_1 or not.

Down neighbor:

if $(y_2 + s_2 = y_1)$

if $(x_2 < x_1)$

if $(x_2 + s_2 > x_1)$, then n_2 is the **down** neighbor else n_2 is not the neighbor.

else

if $(x_2 < x_1 + s_1)$, then n_2 is the **down** neighbor else n_2 is not the neighbor.

Up neighbor:

elseif $(y_2 = y_1 + s_1)$

if $(x_2 < x_1)$

if $(x_2 + s_2 > x_1)$, then n_2 is the **up** neighbor else n_2 is not the neighbor.

else

if $(x_2 < x_1 + s_1)$, then n_2 is the **up** neighbor else n_2 is not the neighbor.

Left neighbor:

elseif $(x_2 + s_2 = x_1)$

if $(y_2 < y_1)$

if $(y_2 + s_2 > y_1)$, then n_2 is the **left** neighbor else n_2 is not the neighbor.

else

if $(y_2 < y_1 + s_1)$, then n_2 is the **left** neighbor else n_2 is not the neighbor.

Right neighbor:

elseif $(x_2 = x_1 + s_1)$

if $(y_2 < y_1)$

if $(y_2 + s_2 > y_1)$, then n_2 is the **right** neighbor else n_2 is not the neighbor.

else

if $(y_2 < y_1 + s_1)$, then n_2 is the **right** neighbor else n_2 is not the neighbor.

else n_2 is **not** the neighbor.

B. Image Model and Oracle R-D Performance

We consider the polygonal model, where there is a white polygon-shaped object with V vertices against a uniform black background. Assume that the image is defined on the unit square $[0,1]^2$. In such a case, a possible oracle method simply codes the position of the V vertices of the polygon. With R/V bits for each vertex, a regular grid on the unit square provides quantized points within a distance $\Delta = \frac{1}{\sqrt{2}} 2^{-\frac{R}{2V}}$ from the original vertices. As each side-length of the polygon is bounded by $\sqrt{2}$ (the diagonal of the unit square), the total length of the boundary of the polygon is bounded by $\sqrt{2}V$. Hence, the distortion for the 2-D object is upper bounded by $D(R) \leq \sqrt{2}V\Delta$. Therefore, for the polygonal model, the oracle R-D function decays exponentially as

$$D(R) \leq V 2^{-R/2V}. \tag{21}$$

In the next two Sections III-C and III-D, we present the R-D performance of the two quadtree algorithms for the polygonal model. This analysis can be extended to the more general piecewise polynomial image model, where the edge is also a piecewise polynomial curve [24].[11]

C. R-D analysis of the prune quadtree algorithm

Similar to the 1-D case, first we show that the prune quadtree scheme encodes a number of leaves, which increases linearly with respect to the tree depth J. We then present Theorem 3, which states the suboptimal R-D behavior of the prune quadtree scheme.

Lemma 3: The prune quadtree coding algorithm will result in a quadtree with a number of leaves upper-bounded by $(3J+1)\,V$, where J and V represent the decomposition depth of the tree and the number of vertices of the polygon in the image, respectively.

Proof: Similar to the 1-D scenario, at high rates, the prune quadtree segmentation scheme recursively divides only those dyadic blocks which contain a vertex of the polygon edge. Other dyadic blocks contain either no edge or a straight edge, so they can be efficiently represented by the edge tiles. Since the polygon has V vertices, there are at most V splitting nodes at each tree

[11]For piecewise polynomial images with piecewise polynomial boundaries, the quadtree algorithm uses edge tiles which consist of two 2-D polynomials separated by a polynomial boundary.

level. Thus, they will generate no more than $3V$ leaves with a straight edge at the next level. The leaves generated at depth J will be $4V$, while the level 0 cannot have any leaf at high rates for $V > 0$. Hence, the total number N_0 of leaves in the pruned quadtree is bounded as follows

$$N_0 \quad \leq \quad (J-1)3V + 4V = (3J+1)\,V. \tag{22}$$

◇

Similar to the 1-D case, every tree level can have at most $4V$ nodes. Therefore, the total number M_0 of nodes in the pruned quadtree can be given by

$$M_0 \quad \leq \quad 4JV + 1. \tag{23}$$

The polygonal model image has a finite number of degrees of freedom, while the prune quadtree scheme codes a number of parameters which grows linearly with J. Therefore, it is bound to exhibit a suboptimal R-D behavior. This is more formally enunciated in the following theorem, which we do not prove here since the proof follows the same logic of Theorem 1.

Theorem 3: For the polygonal model, the prune quadtree coding algorithm, which employs the parent-children pruning, results in the following asymptotic R-D behavior

$$D_P\,(R) \leq c_6 \sqrt{R} 2^{-c_7\sqrt{R}}, \tag{24}$$

where $c_6 = 2\sqrt{6V}$ and $c_7 = \sqrt{\frac{2}{3V}}$.

D. R-D analysis of the prune-join quadtree algorithm

In this section, we show that the neighbor joint coding strategy leads to the desired exponentially decaying R-D behavior. First of all, by following the same steps of Lemma 2, one can prove the following lemma:

Lemma 4: The prune-join quadtree algorithm, which jointly encodes similar neighbors, reduces the effective number of leaves to be encoded to V, where V is the number of vertices of the polygon in the image.

Proof: Similar to the 1-D case, it is obvious that the two neighboring leaves will be joined to improve the R-D performance, if the joint block can be well represented by an edge tile. It is also clear that there will be at most V leaves with vertices at the tree depth J. If J is large enough, then in the worst case each vertex will lie in a different dyadic square leaf. Hence, V leaves cannot be represented by the edge tiles. Since the image can be characterized by only V vertices, only V different linear pieces exist in the image. Therefore, only V edge tiles can have different linear pieces. Similar to the 1-D case, the neighbor joint coding ensures that all the similar leaves characterized by same linear piece will be joined to form one joint block. Since the image has V

different linear pieces, the neighbor joint coding will result into V joint blocks. Therefore, the prune-join tree algorithm provides V joint leaves and V leaves with a vertex. Since the leaves containing a vertex will not be coded, the number of the encoded leaves becomes V. $\diamond$

We are now in the position to state the following theorem:

Theorem 4: For the polygonal model, the prune-join quadtree algorithm, which jointly encodes similar neighbors, achieves the oracle like exponentially decaying asymptotic R-D behavior

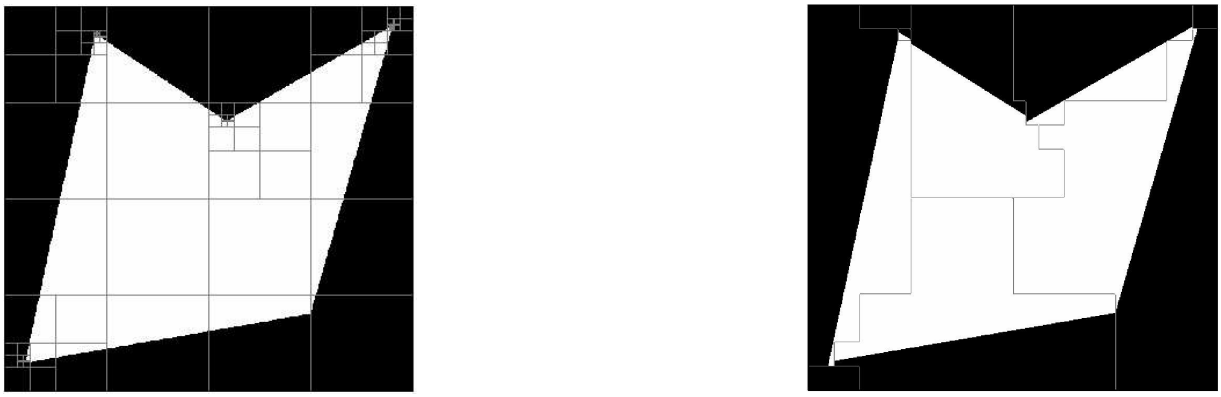
$$D_{PJ}(R) \leq c_8 2^{-c_9 R}, \tag{25}$$

where $c_8 = \frac{5}{2}V$ and $c_9 = \frac{2}{17V}$.

Proof: The prune-join quadtree algorithm provides V joint blocks to be encoded. In the worst case, each vertex will lie in a different dyadic leaf at the depth J. Their sizes will be 2^{-2J}. Therefore, the squared error distortion of each of the vertex containing leaves is bounded by $\frac{1}{4}2^{-2J}$, if the node is represented by the mean value $\frac{1}{2}$ of the image dynamic range $(0, 1)$. For coding the joint block with a linear-edge, we need to code the locations of two vertices of the linear-edge on the boundary of the unit square. The encoding order of these two vertices is simply used to specify the value of the associated regions: for example, when one traverses from first vertex to the second one, the black region is on the left. In this case, if we allocate r bits to each line-vertex of the linear edge of a joint leaf, then the maximum distance between the true line vertices and their quantized version is bounded by $2^{-(r-1)}$. Thus, the distortion of the joint leaf will be bounded by $2^{-(r-1)}$, and this distortion bound will be achieved if the linear edge is the diagonal of the unit square. Similar to 1-D, R-D optimization results in a tree-depth J and a bit allocation strategy such that the joint leaves and the vertex containing leaves have a distortion of the same order $O\left(2^{-2J}\right)$. Therefore, the coding scheme will allocate no bits to leaves with vertices and $2(2J+1)$ bits to every joint block having a linear piece of the polygonal edge to ensure that the distortion for each joint leaf is bounded by 2^{-2J}.[12] As there are only V joint leaves, the bitrate required for coding the leaves is $R_{\text{Leaves}} = 2(2J+1)V$.

The bitrate R_{Tree} needed for coding the quadtree structure is equal to the total number of nodes in the pruned tree. Thus, (23) provides $R_{\text{Tree}} \leq 4JV + 1$. For coding the neighbor joint coding information, we need at most three bits for each leaf as the first bit indicates the joint coding decision and the next two bits provide the neighbor index. Thus, the bitrate needed to code the leaf joint coding information is $R_{\text{LeafJointCoding}} \leq 3 \cdot (3J+1)V$ (from (22)). Hence, the

[12]Each line-vertex is coded using $2J+1$ bits.



(a) Prune tree segmentation. (b) Prune-join tree segmentation.

Fig. 9. Examples of the quadtree representation for the polygonal model.

total bitrate is as follows

$$\begin{aligned} R &= R_{\text{Tree}} + R_{\text{LeafJointCoding}} + R_{\text{Leaves}} \\ R &\leq 17\left(J + \frac{1}{2}\right)V; \text{ as } V > 0. \end{aligned} \tag{26}$$

The net distortion is the sum of the distortions of V joint leaves and V leaves with a vertex and it can be expressed as follows

$$D_{PJ} = V\left(2^{-2J}\right) + V\left(\frac{1}{4}2^{-2J}\right) \tag{27}$$

Combining (26) and (27) provides

$$D_{PJ} \leq \frac{5}{2}V2^{-\frac{2}{17V}R}. \tag{28}$$

Therefore, the prune-join tree algorithm achieves an exponentially decaying R-D behavior. □

An example of the two schemes is shown in Figure 9. It is also of interest to note that the prune-join scheme captures a complex geometrical tiling of an image without any significant increase in the complexity.

E. Computational complexity

The main difference between the binary tree and quadtree algorithm is that the quadtree scheme employs more complex geometrical edge tiles. Unlike 1-D, we can approximate a quadtree node either by a polynomial model (smooth model) or by a piecewise polynomial model with a linear edge (edge model). Consider an image of size $n \times n$. The quadtree decomposition is performed up to the maximum tree depth $\widehat{J} = \log n$. Thus, the total number of nodes will be $O(n^2)$ and the average node size will be $O(\log n)$.

• *Smooth models*: Similar to the 1-D case, we need to follow the Vandermonde matrix based approach for computing the best 2-D Legendre polynomial approximation for a tree node. In 2-D, a P^{th} order polynomial $p(x,y)$ over a region Ω is defined as follows:[13]

$$p(x,y) = \sum_{i=0}^{P}\sum_{j=0}^{P-i} a_{ij}x^i y^j = \sum_{i=0}^{P}\sum_{j=0}^{P-i} l_{ij}\phi_{ij}(x,y),$$

where $\phi_{ij}(x,y), 0 \leq i,j,i+j \leq P$, are the 2-D Legendre polynomial basis functions over the region Ω and $l_{ij}, 0 \leq i,j,i+j \leq P$, are the associated Legendre polynomial coefficients. Similar to the 1-D case, 2-D Legendre polynomial basis functions are computed by applying the Gram-Schmidt orthogonalization procedure on the standard polynomial basis set $\{x^i y^j, 0 \leq i,j,i+j \leq P\}$. For example, if the underlying region $\Omega = (-1,1)\times(-1,1)$, then $\phi_{00} = \frac{1}{2}, \phi_{01} = \frac{\sqrt{3}}{2}y, \phi_{10} = \frac{\sqrt{3}}{2}x$. Now, in the discrete set-up, for a 2-D segment $\boldsymbol{Z}$ of size L (total number of pixels) with the underlying column ordered grid $\widehat{\Omega} = \{(x_k, y_k), 1 \leq k \leq L\}$,[14] the minimum squared-error Legendre polynomial approximation $\boldsymbol{p}$ of order P is obtained by solving the least square (LS) problem:

$$\min_{\boldsymbol{p}} \|V_{L,P}\,\boldsymbol{p} - \boldsymbol{z}\|^2, \tag{29}$$

where $\boldsymbol{p}$ is a vector of $\frac{(P+1)(P+2)}{2}$ polynomial coefficients, $\boldsymbol{z}$ is the column ordered form of the 2-D segment $\boldsymbol{Z}$, and $V_{L,P}$ is the following $L \times \frac{(P+1)(P+2)}{2}$ Vandermonde matrix:

$$V_{L,P} = \begin{bmatrix} \phi_{00}(x_1,y_1)\cdots\phi_{0P}(x_1,y_1)\cdots\phi_{m0}(x_1,y_1)\cdots\phi_{m(P-m)}(x_1,y_1)\cdots\phi_{P0}(x_1,y_1) \\ \phi_{00}(x_2,y_2)\cdots\phi_{0P}(x_2,y_2)\cdots\phi_{m0}(x_2,y_2)\cdots\phi_{m(P-m)}(x_2,y_2)\cdots\phi_{P0}(x_2,y_2) \\ \cdots \\ \phi_{00}(x_L,y_L)\cdots\phi_{0P}(x_L,y_L)\cdots\phi_{m0}(x_L,y_L)\cdots\phi_{m(P-m)}(x_L,y_L)\cdots\phi_{P0}(x_L,y_L) \end{bmatrix} \tag{30}$$

Similar to 1-D, the solution to the LS problem in (29) is attained efficiently by means of a QR factorization of $V_{L,P}$ with computational cost of $O\left(L\frac{(P+1)(P+2)}{2}\right)$. The Vandermonde matrix $V_{L,P}$ basically depends on the underlying grid, which is same for all the nodes at the same tree level as all nodes of a tree level are of the same size. Thus, only one Vandermonde matrix is required per tree level to compute smooth models. Therefore, we can pre-compute and store these matrices and their QR factorization for different tree levels and use them for computing 2-D polynomials for tree nodes just like a look-up table. Since the average node-size is $O(\log n)$, the overall cost for computing the smooth models for all the tree nodes will be $O(n^2 \log n)$. Note that for the complexity analysis, we include $\frac{(P+1)(P+2)}{2}$ in the complexity constant.

• *Edge models*: These are represented by two 2-D polynomials separated by a linear boundary. Therefore, for each node, we need to search for the best edge model for a given set of edge

[13]Note that the P^{th} order 2-D polynomial is defined by $\frac{(P+1)(P+2)}{2}$ coefficients.

[14]where k is the 1-D index obtained by column ordering the 2-D grid like MATLAB.

TABLE I

SUMMARY OF THE PROPERTIES OF THE DIFFERENT ALGORITHMS.

	R-D behavior			
Signal class	Wavelet coder	DP coder	Prune tree coder	Prune-join tree coder
1-D PPS	$d_0\sqrt{R}2^{-d_1\sqrt{R}}$	$c_0 2^{-c_1 R}$	$c_2\sqrt{R}2^{-c_3\sqrt{R}}$	$c_4 2^{-c_5 R}$
2-D Polygonal	$d_2 \log R \cdot R^{-1}$ [6]	NA	$c_6\sqrt{R}2^{-c_7\sqrt{R}}$	$c_8 2^{-c_9 R}$
	Computational cost			
1-D PPS	$O\left(NR_Q \log N\right)$	$O\left(N^3 R_Q\right)$	$O\left(NR_Q \log N\right)$	$O\left(NR_Q \log N\right)$
2-D Polygonal	$O\left(NR_Q \log N\right)$	NA	$O\left(NR_Q \log N\right)$	$O\left(NR_Q \log N\right)$

orientations like the wedgelet/beamlet dictionary [9].[15] Thus, for each edge-orientation, we need two Vandermonde matrices associated with the two regions separated by the edge. We can pre-compute these Vandermonde matrices as given by (30). Now, we can compute the best polynomial surfaces associated with each choice of edge orientation using the Vandermonde matrix approach. We then select that edge orientation which leads to the minimum squared error. The edge orientation dictionary and associated Vandermonde matrices are pre-computed and stored so that the algorithm can use them like a look-up table.[16] Since the average node size is $O(\log n)$, the computational cost for calculating the edge model for a tree node is $O(\log n)$. Hence, the overall cost of computing the edge models for all the tree nodes will be $O(n^2 \log n)$.

For an image of size $n \times n$, the total number of pixels is $N = n^2$. Suppose that R_Q quantizers are utilized for the R-D function computation. Now, by following the steps of the computational analysis done in Section II-E for the 1-D case, it can be shown that the overall computational costs for both the prune and the prune-join quadtree algorithms will be $O\left(NR_Q \log N\right)$. Table I summarizes the properties of the tree algorithms and compares them with a wavelet coder and a dynamic programming (DP) coder. But note that DP is not applicable (NA) in the 2-D case.

[15]To achieve the theoretical R-D performance for the polygonal model, the algorithm uses the edge-dictionary with $O(m^2 \log m)$ linear-edge orientations for a node of size $m \times m$, where m^2 is, on average, $O(\log n)$. This is also consistent with the high rate analysis. Moreover, by using the side information that the polygonal image is binary, the algorithm sets $P = 0$ and codes only the linear-edge and the constant value (1 or 0) above the linear-edge to efficiently code a node. However, for real images, we limit the maximum number of linear edge choices in the edge-dictionary to 256, irrespective of the image-size. This is similar as saying that the linear edge is quantized using no more than 8 bits.

[16]Note that the storage memory requirement is proportional to the size of the edge-orientation dictionary.

IV. Simulation results and discussion

A. 1-D case

In this numerical experiment, we consider piecewise quadratic polynomials with no more than $S = 32$ singularities. Polynomial coefficients and singular points are generated randomly using the uniform distribution on the range $[-1, 1]$. The Legendre polynomial coefficients associated with a node are scalar quantized with different quantizers. The tree scheme chooses eight possible quantizers operating at rates $4, 8, 12, 16, 20, 24, 28$ and 32 bits. The algorithm also needs to code the selected quantizer choice using 3 bits as the side information.

In Figure 10, we compare R-D performance of the two proposed binary tree coding algorithms against their theoretical R-D behaviors. Figure 10 shows that the R-D behaviors of the two coding schemes are consistent with the theory. In particular, the prune-join binary tree algorithm achieves the exponentially decaying R-D behavior.

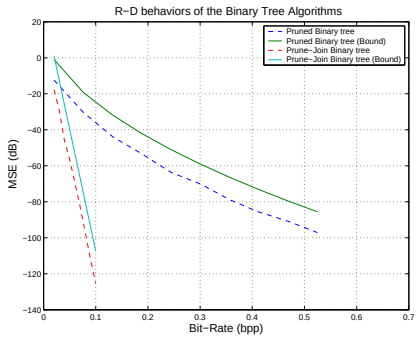


Fig. 10. Theoretical (solid) and numerical (dotted) R-D curves for the prune and prune-join binary tree algorithms for piecewise polynomial signals.

B. 2-D case

Numerical experiments are performed for two image classes: 1) Polygonal model, where the polygon's vertices are generated randomly using uniform distribution on the space $[0, 1]^2$. 2) Real images.

For the polygonal images, the edge-tile is simply composed of two constant regions separated by a linear edge. However, for real images, an edge-tile is composed of two 2-D polynomials, of degree $\leq P$, separated by a linear boundary. Hence, the algorithm can represent any surface by one of the $P+1$ polynomial models. For real images, our scheme allows for up to piecewise quadratic models ($P = 2$). Therefore, any surface can be approximated by either constant or linear or quadratic polynomial model. Thus, the algorithm will compute $(P + 1)$ smooth and $(P + 1)^2$ edge models for each tree node.[17] For the given bit budget, the algorithm selects the model with minimum

[17]Since an edge model is composed of two surfaces and each surface can select any one of the $(P + 1)$ polynomial models, there are $(P + 1)^2$ possible edge models.

Lagrangian cost for a node. This model choice is coded using $\lceil \log_2 \left((P+1) + (P+1)^2 \right) \rceil$ bits as a side information. For $P = 2$, the algorithm uses 4 bits to indicate the model choice.

For synthetic piecewise polynomial images, we simply use the uniform scalar quantizer to code the 2-D Legendre polynomial coefficients. But, for real images, we need to use the non-uniform quantizer [13] for coding the higher order polynomial coefficients, as higher order polynomial coefficients seem to have Laplacian like distribution. However, the zeroth order coefficient is always coded using the uniform quantizer. The edge-orientation choice is coded by its index in the edge-dictionary, which basically represents the quantization of edge-orientations.

It is obvious that the higher order polynomial models should perform better from the non-linear approximation point of view. However, when the goal is compression, then the answer is not simple as coding of higher order polynomial may require a large increase in rate without significant reduction in the overall distortion. That is why our scheme selects the appropriate polynomial/edge tile according to the Lagrangian cost based R-D criterion to achieve better R-D performance. Simulation results shown in Figure 12 indicate that the algorithm opts for low order polynomial models at low rates.

For the cameraman image, simulation results show that our scheme prefers piecewise linear model over piecewise quadratic model at rates less than 0.2 bpp. Even at higher rates, we gain slightly by using piecewise quadratic models. Thus, the piecewise linear polynomial model seems to be a good modeling choice for cameraman at low rates. Finally, in simulations, we have used only linear boundary model, which is a good model for edges at low rates.

The experimental results shown in Figure 11, for the polygonal model, confirm the derived theoretical R-D behaviors. Figure 12 shows the complex geometrical tiling obtained by the prune-join tree scheme for the cameraman image. In Figures 13, 15 and 16, we compare the prune-join coding scheme with JPEG2000. Residual images shown in Figure 14 also demonstrate that the prune-join scheme captures the image geometry more efficiently in comparison to JPEG2000. Figure 14(a) also shows that the residual image obtained by the tree coding scheme essentially contains only the texture part of the cameraman image. Figure 17 compares the prune-join scheme with JPEG2000 for different regions of the lena image. When the image is close to the geometrical model (see Figures 17 (a), (b)), the prune-join scheme gives less artifacts. In the textured region (see Figures 17 (c), (d)), the geometrical model fails, and JPEG2000 performs better. Overall, these simulation results indicate that the prune-join coding scheme attains not only better visual quality but also higher coding gain in comparison to JPEG2000. Moreover, Table II shows that the prune-join tree algorithm consistently outperforms both the prune tree algorithm and JPEG2000 for different real images at low bit rates. It does so particularly well for

the cameraman image compared to the other images. One possible reason is that the cameraman image is much closer to the piecewise polynomial image model in comparison to the other images.

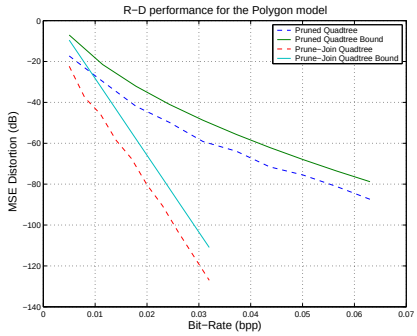


Fig. 11. Theoretical (solid) and numerical (dotted) R-D curves for the prune and prune-join quadtree algorithms for the polygonal image class.

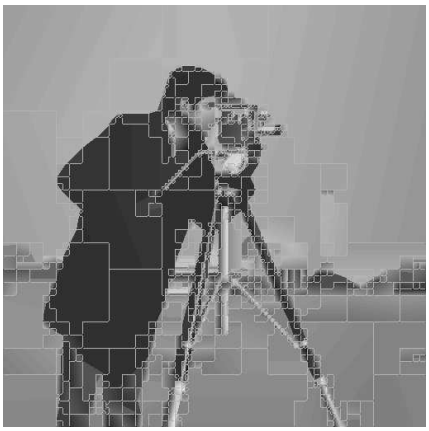


Fig. 12. Prune-join quadtree tiling for the cameraman image at bitrate=0.071 bpp.

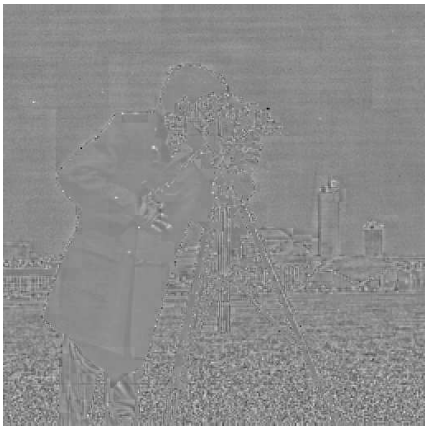
V. Conclusions

For 1-D piecewise polynomials, we have presented an efficient binary tree based compression algorithm, which achieves oracle like exponentially decaying R-D behavior with low computational cost of $O(N \log N)$. Similar R-D performance can also be achieved by the dynamic segmentation algorithm proposed in [19] with large computational cost of $O\left(N^3\right)$. The dynamic programming techniques cannot be extended to the 2-D case, whereas our binary tree based coding algorithm can be extended to the 2-D case in the form of quadtree based coding algorithm with low computational complexity of $O(N \log N)$. We have also proved that the quadtree based coding algorithm achieves exponentially decaying asymptotic R-D behavior for the polygonal image model. Numerical simulations (Figure 11) also confirm that the algorithm achieves optimal performance if the input image fits the model exactly. In addition, simulations show

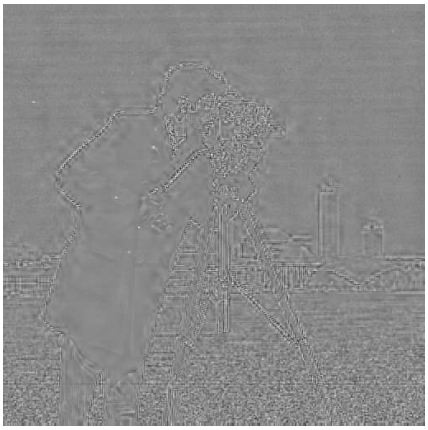
(a) Prune-join quadtree (Rate=0.15 bpp, PSNR=30.68dB).

(b) JPEG2000 (Rate=0.15 bpp, PSNR= 29.21 dB).

Fig. 13. Comparison of the quadtree coder and a wavelet coder (JPEG2000) for the cameraman image.



(a) Prune-join quadtree.



(b) JPEG2000.

Fig. 14. Residual images of the quadtree coder and JPEG2000 for the cameraman image at 0.15 bpp.

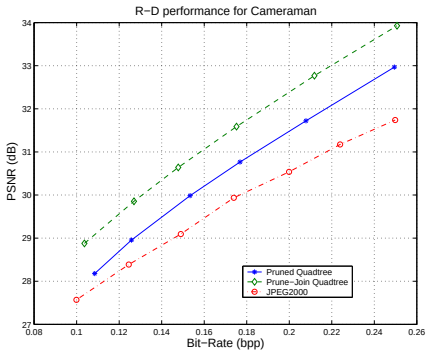


Fig. 15. R-D performance comparison of the quadtree schemes and JPEG2000 for the cameraman image.

(a) Prune-join quadtree (Rate=0.15 bpp, PSNR= 30.86 dB).

(b) JPEG2000 (Rate=0.15 bpp, PSNR= 30.34 dB).

Fig. 16. Comparison of the quadtree coder and a wavelet coder (JPEG2000) for the lena image.

(a) Prune-join scheme.

(b) JPEG2000.

(c) Prune-join scheme.

(d) JPEG2000.

Fig. 17. Comparison of artifacts in two regions of the lena image at 0.15 bpp for the prune-join scheme and JPEG2000.

TABLE II

R-D PERFORMANCE COMPARISON OF DIFFERENT ALGORITHMS FOR DIFFERENT IMAGES.

Image	Bit-rate	PSNR (dB)		
		Prune tree	JPEG2000	Prune-join tree
Cameraman	0.15 bpp	29.85 dB	29.21 dB	30.68 dB
	0.20 bpp	31.45 dB	30.54 dB	32.38 dB
	0.25 bpp	32.98 dB	31.74 dB	33.91 dB
Lena	0.15 bpp	29.48 dB	30.34 dB	30.86 dB
	0.20 bpp	30.95 dB	31.51 dB	31.98 dB
	0.25 bpp	32.31 dB	32.46 dB	33.01 dB
Peppers	0.15 bpp	31.31 dB	32.32 dB	32.81 dB
	0.20 bpp	32.93 dB	33.63 dB	34.04 dB
	0.25 bpp	34.25 dB	34.89 dB	35.16 dB

that our quadtree algorithm consistently outperforms JPEG2000 also in case of compression of real images (Figures 13, 15, 16 and Table II).

REFERENCES

[1] E. J. Candes and D. L. Donoho, "Curvelets- a surprisingly effective non-adaptive representation for objects with edges," in *Curve and Surface Fitting*, A. Cohen, C. Rabut, and L. L. Schumaker, Eds., Saint-Malo, 1999, Vanderbilt University Press.

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

[3] A. Cohen, I. Daubechies, O.G. Guleryuz and M.T. Orchard, "On the importance of combining wavelet-based nonlinear approximation with coding strategies," *IEEE Trans. Inform. Th.,*, vol. 48, pp. 1895 -1921, July 2002.

[4] A.Cohen and B.Matei, "Compact representations of images by edge adapted multiscale transforms," in *Proc. IEEE Int. Conf. on Image Proc.(ICIP)*, Thessaloniki, Greece, Oct. 2001.

[5] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, John Wiley and Sons, 1991.

[6] M.N. Do, P.L. Dragotti, R. Shukla and M. Vetterli, "On the compression of two dimensional piecewise smooth functions," in *Proc. IEEE Int. Conf. on Image Proc.(ICIP)*, Thessaloniki, Greece, Oct. 2001.

[7] M. N. Do and M. Vetterli, "Contourlets: Beyond Wavelets," J. Stoeckler and G. V. Welland eds., Academic Press, 2003.

[8] D.L. Donoho, "Wedgelets: Nearly minimax estimation of edges", *Annals of Statistics*, 27(3):859-897, 1999.

[9] D.L. Donoho and X. Huo, "Beamlets and multiscale image analysis," Tech. Rep., Department of Statistics, Stanford University, 2001.

[10] P.L. Dragotti and M. Vetterli, "Wavelet footprints: theory, algorithms and applications," *IEEE Trans. Signal Proc.*, vol. 51(5), pp. 1306-1323, May 2003.

[11] V. K. Goyal, "Theoretical foundations of transform coding," *IEEE SP Mag.*, vol. 18, no. 5, pp. 9-21, Sep. 2001.

[12] J. J. Y. Huang and P. M. Schultheiss, "Block quantization of correlated Gaussian random variables," *IEEE Trans. Comm.*, vol. 11, pp. 289-296, Sep. 1963.

[13] A.K. Jain, *Fundamentals of Digital Image Processing*, Prentice Hall Inc., 1989.

[14] W. S. Lee, "Tiling and adaptive image compression," *IEEE Trans. on Inform. Th.*, vol. 46, no. 5, Aug. 2000.

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

[16] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, San Diego, CA, USA, 1997.

[17] W.B. Pennebaker and J.L.Mitchell, *JPEG Still Image Data Compression Standard*, Van Nostrand Reinhold, New York, 1993.

[18] E. Le Pennec and S. Mallat, "Bandelet representation for image compression," in *Proc. IEEE Int. Conf. on Image Proc.(ICIP)*, Thessaloniki, Greece, Oct. 2001.

[19] P.Prandoni and M.Vetterli, "Approximation and compression of piecewise smooth functions," *Phil. Trans. Royal Society London*, vol. 357, no. 1760, pp. 2573-2591, Sep. 1999.

[20] P. Prandoni, *Optimal Segmentation Techniques for Piecewise Stationary Signals*, PhD Thesis, Swiss Federal Institute of Technology, Lausanne, Switzerland, 1999.

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

[22] H. Radha, M. Vetterli and R. Leonardi, "Image compression using binary space partitioning trees," *IEEE Trans. Image Proc.*, vol. 5, no. 12, pp. 1610-1624, Dec. 1996.

[23] K.Ramchandran and M.Vetterli, "Best wavelet packet bases in a rate distortion sense," *IEEE Trans. Image Proc.*, vol. 2, no. 2, pp. 160-175, April 1993.

[24] R. Shukla, *Rate-distortion optimized geometrical image processing*, PhD thesis submitted to Swiss Federal Institute of Technology, Lausanne, Switzerland, March 2004.

[25] R. Shukla, P.L. Dragotti, M.N. Do and M. Vetterli, "Improved quadtree algorithm based on joint coding for piecewise smooth image compression," in *Proc. IEEE Int. Conf. on Multimedia*, Switzerland, Aug. 2002.

[26] "Transform Coding: Past, Present and Future," Special Issue, *IEEE SP Mag.*, vol. 18, no. 5, Sep. 2001.

[27] G.J. Sullivan and R.L. Baker, "Efficient quadtree coding of images and video," *IEEE Trans. Image Proc.*, vol. 3, no. 3, pp. 327-331, May 1994.

[28] D. Taubman and M. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards and Practice*, Kluwer Academic Publishers, Dordrecht, 2001.

[29] D.J. Vaisey and A. Gersho, "Image coding with variable block size segmentation," *IEEE Trans. Signal Proc.*, vol. SP-40, pp. 2040-2060, Aug.1992.

[30] R. M. F. Ventura, L. Granai and P. Vandergheynst, "R-D analysis of adaptive edge representations," in *Proc. of IEEE MMSP*, Dec. 2002.

[31] M.Vetterli, "Wavelets, approximation and compression," *IEEE SP Mag.*, vol. 18, no. 5, pp. 59-73, Sep. 2001.

[32] M. Wakin, J. Romberg, H. Choi, and R. Baraniuk, "Rate-distortion optimized image compression using wedgelets," in *Proc. IEEE Int. Conf. on Image Proc.(ICIP)*, USA, Sep. 2002.

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