RATE-DISTORTION OPTIMIZED GEOMETRICAL IMAGE PROCESSING

THÈSE N^{O} 2992 (2004)

PRÉSENTÉE À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

Institut de systèmes de communication

SECTION DES SYSTÈMES DE COMMUNICATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Rahul SHUKLA

Bachelor of Technology in Electrical Engineering, Indian Institute of Technology, Kanpur, Inde
et de nationalité indienne

acceptée sur proposition du jury:

Prof. M. Vetterli, directeur de thèse
Prof. M. Do, rapporteur
Dr P. Dragotti, rapporteur
Prof. H. Radha, rapporteur
Prof. M. Unser, rapporteur

Lausanne, EPFL
2004

Abstract

Since geometrical features, like edges, represent one of the most important perceptual information in an image, efficient exploitation of such geometrical information is a key ingredient of many image processing tasks, including compression, denoising and feature extraction. Therefore, the challenge for the image processing community is to design efficient geometrical schemes which can capture the intrinsic geometrical structure of natural images.

This thesis focuses on developing computationally efficient tree based algorithms for attaining the optimal rate-distortion (R-D) behavior for certain simple classes of geometrical images, such as 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. We first investigate both the one dimensional (1-D) and two dimensional (2-D) piecewise polynomials signals. For the 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 and is called prune-join algorithm. This allows to achieve the correct exponentially decaying R-D behavior, $D(R) \sim 2^{-cR}$, thus improving over classical wavelet schemes. We also show that the computational complexity of the scheme is of $O(N \log N)$. We then extend this scheme to the 2-D case using a quadtree, which also achieves an exponentially decaying R-D behavior, for the piecewise polynomial image model, with a low computational cost of $O(N \log N)$. Again, the key is an R-D optimized prune and join strategy.

We further analyze the R-D performance of the proposed tree algorithms for piecewise smooth signals. We show that the proposed algorithms achieve the oracle like polynomially decaying asymptotic R-D behavior for both the 1-D and 2-D scenarios. Theoretical as well as numerical results show that the proposed schemes outperform wavelet based coders in the 2-D case.

We then consider two interesting image processing problems, namely denoising and stereo image compression, in the framework of the tree structured segmentation. For the denoising problem, we present a tree based algorithm which performs denoising by compressing the noisy image and achieves im-

proved visual quality by capturing geometrical features, like edges, of images more precisely compared to wavelet based schemes. We then develop a novel rate-distortion optimized disparity based coding scheme for stereo images. The main novelty of the proposed algorithm is that it performs the joint coding of disparity information and the residual image to achieve better R-D performance in comparison to standard block based stereo image coder.

Résumé

Puisque les éléments géométriques, comme les bords, représentent des information perceptuelles parmi les plus importantes dans une image, l'exploitation efficace de telles information géométriques dans les images est un ingrédient principal de nombreuses applications du traitement d'image, y compris la compression, la réduction du bruit et l'extraction de caractéristiques. Par conséquent, le défi pour la communauté du traitement d'image consiste à concevoir des méthodes géométriques efficaces capables de discerner la structure géométrique intrinséque des images naturelles.

Cette thèse se concentre sur le développement d'algorithmes efficaces basés sur des arbres pour atteindre le comportement optimal de la fonction rate-distortion (R-D) pour certaines classes simples d'images géométriques telles que les images polynômiales par morceaux avec des frontières polynômiales. Une bonne approximation de cette classe permet de développer des méthodes d'approximation et de compression efficaces pour des images avec de fortes caractéristiques géométriques et également, comme les résultats expérimentaux le montrent, pour des images naturelles. Nous étudions d'abord les signaux polynômiaux par morceaux aussi bien dans une dimension (1-D) que dans deux dimensions (2-D). Pour le cas 1-D, notre méthode est basée sur la segmentation du signal par arbre binaire. Cette méthode fait une approximation des segments du signal en utilisant des modèles polynomiaux ainsi qu'une stratégie d'attribution des bits aux différents segments du signal qui est optimale au sens de R-D. En outre, cette méthode code des voisins similaires conjointement et est appelée algorithme tailler-joindre "prune-join". Ceci nous permet d'obtenir le comportement de décroissance exponentielle correct au sens de R-D, $D(R) \sim 2^{-cR}$, et de ce fait de surpasser les méthodes classiques par ondelettes. Nous prouvons également que la complexité de la méthode est de $O(N \log N)$. Nous généralisons ensuite cette méthode pour le cas 2-D en utilisant un 'quadtree', ce qui conduit aussi à une décroissance exponentielle de la fonction R-D pour des images polynômiales par morceaux, avec une complexité faible de $O(N \log N)$. Là encore, la solution consiste en une stratégie de joindre et tailler optimisée au sens de R-D.

De plus nous analysons l'efficacité R-D des algorithmes par arbres proposés pour des signaux régulier par morceaux. Nous montrons que les algorithmes pro-

posés permettent d'obtenir le comportement asymptotique R-D de décroissance polynômiale, comme celui en présence d'un oracle, aussi bien pour le scénario à une dimension que pour celui à deux dimensions. Des résultats théoriques ainsi que numériques montrent que les méthodes proposées surpassent les codeurs par ondelettes pour le cas 2-D.

Nous considérons ensuite deux problèmes intéressants de traitement d'image, à savoir la réduction du bruit et la compression d'images stéréo, dans le cadre de la segmentation par structures d'arbre. Pour le problème de la réduction du bruit, nous présentons un algorithme par arbre qui effectue la réduction du bruit en comprimant l'image bruitée et permet d'obtenir une qualité visuelle améliorée en discernant des éléments géométriques, comme les bords, dans les images avec plus de précision que les méthodes par ondelettes. Nous développons ensuite une méthode de codage innovatrice basée sur la disparité et optimisée au sens de R-D. La nouveauté principale de l'algorithme proposé consiste dans le fait qu'il effectue le codage de l'information de disparité et de l'image résiduelle conjointement pour atteindre une efficacité R-D supérieure par rapport aux codeurs classiques d'images stéréo par blocs.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to sincerely thank my advisor Martin Vetterli for giving me an opportunity to work on an interesting problem along with the continuous motivation to finish it. I am also greatly indebted to Pier Luigi Dragotti and Minh Do for their continuous collaboration and guidance, which really help me to complete this thesis on time. I would also like to thank Hayder Radha for introducing me to the world of stereo images, which helps me to understand the 3-D world better. All of them have had a profound impact on my understanding of how to approach and solve research problems.

I thank all my fellow graduate students and colleagues at EPFL for their generous help and fruitful discussions, particularly Razvan Cristescu, Christof Faller, Vladan Velisavljevic, Robert Lee Konsbruck, Patrick Vandewalle, Thibaut Ajdler, Williams Devadason, Francoise Behn and Jocelyne Plantefol. I am also grateful to my Indian friends, Rajesh, Prasenjit, Debjani, Usha, Anwitaman, Sushil, James and many more, for making my stay at Lausanne comfortable and enjoyable.

Finally, I would like to dedicate this work to my parents, whose constant support, understanding and encouragement have really brought me here.

Chapter 1

Introduction

1.1 Motivation

From the literature, especially image processing [24], computer vision [2] and harmonic analysis [5], it is well known that geometrical features, like edges, represent one of the most important perceptual and objective information in an image. Thus, the precise modeling of the geometrical information is very crucial in several image processing applications, like compression, denoising and computer vision. Since edges themselves exhibit a certain degree of smoothness, an efficient image processing method must be capable of handling the geometrical regularity of images. Now, the key question of our interest is whether the current state of the art wavelet based schemes are able to capture the geometry of an image efficiently. In the following, we try to understand how well or poorly wavelet based schemes perform for the compression problem.

In recent years, wavelets have become central to many signal processing applications, in particular approximation and compression. In the latest wavelet coders and JPEG2000 [61], wavelets are used because of their good non-linear approximation (NLA) properties for piecewise smooth functions in one dimension [67]. In 1-D, wavelets derive their good NLA behavior from their vanishing moment properties [32], and the question now is to see how these properties carry through in the rate-distortion scenario. Even if good approximation properties are necessary for good compression, it might not be sufficient. In particular, in non-linear approximation, the indexing and individual compression of wavelet coefficients might be inefficient. It is shown for a simpler class of signals, namely piecewise polynomials, in [9, 40] that 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 hard to see that the rate-distortion (R-D) behavior of an oracle based method

decays as[1]

$$D(R) \sim c_0 2^{-c_1 R}. \tag{1.1}$$

Thus, even in 1-D, wavelets based schemes perform suboptimally because they fail to precisely model singularities.

In the 2-D scenario, the situation is much worse. The reason is that wavelets in 2-D are obtained by a tensor-product of one dimensional wavelets, so 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, especially for compression.

To understand the magnitude of this limitation, consider a simple 2-D function which is composed of two 2-D polynomials separated by a linear boundary. Assume that we apply the wavelet transform with enough vanishing moments on this function. Then, at a level j, the number of significant wavelet coefficients corresponding to 2-D polynomial regions are bounded by a constant but the number of significant wavelet coefficients representing linear boundary grows exponentially as 2^j. Even if the linear singularity can be represented by only two parameters, namely slope and intercept, the wavelets based scheme models it using an exponentially growing number of wavelet coefficients. This failure to recognize the smoothness of a singularity in 2-D leads to the following suboptimal R-D behavior [14]

$$D(R) \sim \frac{\log R}{R}, \tag{1.2}$$

which is far from the optimal *exponentially decaying* R-D behavior.[2]

Since wavelets cannot efficiently model singularities along lines or curves, wavelet based schemes fail to explore the geometrical structure that is typical in smooth edges of images. Hence, 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 [5], wedgelets [16], beamlets [17], contourlets [15], bandelets [37] and edge adaptive geometrical scheme [10]. Such schemes try to achieve the correct N-term NLA behavior for certain classes of 2-D functions, which can model images. So far, these schemes have not led to precise R-D analysis, which is usually more difficult than NLA analysis.

Moreover, researchers in the wavelet community are also improving wavelet based schemes. In [47], Romberg *et al.* presented the wavelet-domain hidden

[1] In the thesis, we use the term "rate-distortion (R-D)" as a synonym to the term "distortion-rate (D-R)".

[2] In Chapter 4, we show that the algorithm proposed in this work achieves an exponentially decaying R-D behavior for piecewise polynomial images with piecewise polynomial boundaries.

Markov tree model to capture the joint behavior of wavelet coefficients across scales. Dragotti *et al.* [21] proposed a scheme, named wavelet-footprints, to model the exact dependency of wavelet coefficients across scales, showing that wavelet-footprints can provide a sparser representation of piecewise smooth signals in comparison to wavelets.

Let us go back to our simple 2-D function with a linear edge and analyze it from the geometrical point of view. In the spatial domain, this function can be accurately modeled by a simple approximation edge tile which consists of two 2-D polynomials separated by a line. However, this representation amounts to saying that the given image itself is one of the basis functions and, thus, it will necessarily model the given image correctly. Therefore, let us consider more general piecewise polynomial image model with several linear singularities. In that case, if we partition the image such that each segment can be well approximated by a simple edge tile, then we can achieve the desired approximation/compression performance. However, performing such a segmentation remains an open problem and it is well known that the problem of finding an optimal partition is associated with an NP-HARD problem (see the discussion in [13, Section 4] and references therein).

Since both the computational efficiency and precise modeling of geometrical information are key issues in several image processing tasks, we plan to design and study tree based segmentation schemes which model segments by edge tiles to capture the geometry of images. To sum up, this thesis aims to

1. Understand whether tree based coding algorithms can achieve the correct R-D behavior for certain simple classes of geometrical images without sacrificing computational ease.

2. Search for practical methods within the tree structured segmentation framework, which can provide better R-D performance for natural images.

1.2 Related Work

For image coding applications, tree segmentation based schemes have always been popular due to their manageable computational complexity. 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 [29, 52, 60, 65, 74]. Leonardi *et al.* [29] utilized the classic split and merge segmentation techniques to extract image regions and then approximate the contours and image characteristics of those regions. In [28], Lee proposed adaptive rectangular tiling for image compression by using different probability models for compressing different regions of a wavelet subband. Radha *et al.* [44] presented a binary space partitioning tree

coding scheme, which employed parent-children pruning for searching the optimal tree structure. Recently, Wakin *et al.* [70] extended the zerotree based space frequency quantization scheme by adding a wedgelet symbol [16] 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 [66], which employs non-dyadic rectangular partitioning for the image segmentation.

The tree segmentation based schemes considered in [8, 44, 45, 60, 65, 74] employ the parent children pruning to obtain the optimal tree structures for the given bit budget. But they do not attempt to exploit the dependency among the neighboring nodes with different parents. As we see in the next chapter that the parent children pruning strategy may not achieve the correct R-D behavior due to its failure to model the dependency among neighboring nodes.

Recent work closely related to our work is the wedgelets/beamlets based schemes presented in [16, 17]. 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 edges and 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.

1.3 Thesis Outline and Contribution

The main goal of this thesis is to design tree structured segmentation based compression algorithms which can achieve the oracle like R-D performance for some simple classes of images with computational ease.

The search for an answer to this problem begins in the next chapter which focuses on 1-D piecewise polynomial signals and presents prune and prune-join binary tree segmentation based coding algorithms. We analyze the R-D performance and computational complexity of these two coding schemes. In particular, we show that the prune-join tree algorithm, which jointly encodes similar neighbors, achieves the optimal R-D performance with low computational complexity.

In Chapter 3, we consider more general 1-D piecewise smooth signals and attempt to understand whether the proposed binary tree segmentation algorithms can accurately model piecewise smooth signals. Our analysis shows that both the prune and prune-join tree schemes achieve the oracle like polynomially decaying R-D behavior for piecewise smooth signals.

In Chapter 4, we show the extension of the 1-D scheme to 2-D using a quadtree based scheme. We consider a piecewise polynomial image model, where

the edge is also a piecewise polynomial curve. First, we present the oracle R-D performance which decays exponentially. We then derive the asymptotic R-D behaviors of the proposed quadtree schemes. Again, we prove that the prune-join quadtree scheme achieves the oracle like asymptotic R-D performance while the prune quadtree scheme performs suboptimally. Simulation results show that the proposed prune-join quadtree coding scheme consistently outperforms state of the art wavelet based coder (JPEG2000) also in case of compression of real images like cameraman.

Chapter 5 studies the more complex piecewise smooth images and investigates whether the proposed quadtree based segmentation algorithms can achieve the correct R-D behavior for these images. Similar to 1-D, the R-D analysis proves that both the prune and prune-join tree schemes achieve the oracle like R-D behavior which decays polynomially for piecewise smooth images. Experimental results also confirm the derived R-D behaviors of the quadtree algorithms.

In Chapter 6, we focus on two image processing applications, namely denoising and stereo image compression, which require to efficiently exploit the regularity of both singularities and smooth pieces separated by singularities. For the denoising problem, we present a solution based on the proposed tree algorithm and on the key insight that the lossy compression of a noisy signal can provide the filtered/denoised signal. We then address the problem of stereo image compression and propose a novel rate-distortion (R-D) optimized disparity based coding scheme for stereo images. The main novelty of the proposed scheme is that it performs the joint coding of disparity information and the residual image to achieve an improved R-D performance.

Finally, we summarize the key results and discuss future research directions in Chapter 7.

Chapter 2

Binary Tree Segmentation Algorithms for One Dimensional Piecewise Polynomial Signals

2.1 Motivation

Recently, there has been a growing interest in the study of piecewise polynomial functions as an approximation to piecewise smooth functions, which are powerful mathematical objects used to model a great variety of natural phenomena. The central problem is to determine the best way to represent and to code piecewise polynomial signals. For instance, consider the simple piecewise linear signal signal shown in Figure 2.1. Our aim is to answer the following inter-related questions: 1. What is the best representation of this signal? 2. What is the best rate-distortion (R-D) performance achievable by any coding scheme for this signal?

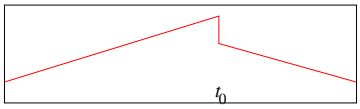


Figure 2.1: A piecewise linear signal with only one discontinuity.

One natural, and possibly the best, way to describe the given signal is as follows: The given signal is a piecewise linear function, which is composed of two linear pieces separated at the discontinuity location t_0. In fact, this is the best representation once we know that the signal belongs to the class of piece-

wise polynomials. 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.[1] Therefore, the oracle method relies on the correct segmentation of the signal for achieving the best R-D performance. Thus, the basic ingredients of the coding scheme are the signal-segmentation and polynomial models applied to the segments.

But, even for this simple piecewise polynomial signal class, the mean squared error (MSE) distortion of wavelet based coders decays as $D(R) \sim d_0 \sqrt{R} 2^{-d_1 \sqrt{R}}$ [9, 40]. The reason is that the wavelet based methods perform independent coding of wavelet coefficients corresponding to a singularity and so they fail to efficiently model singularities. However, In [40], the oracle like R-D behavior has been realized with a polynomial computational cost $\left(O\left(N^3\right)\right)$ using dynamic programming (DP). The dynamic segmentation scheme achieves an exponentially decaying R-D behavior as it performs segmentation according to the break points of the signal and codes only the required number of polynomial pieces. However, due to the exhaustive search for the segmentation, the dynamic programming algorithm is computationally very expensive. The above described example, despite its simplicity, reveals the weaknesses of both wavelet based schemes (suboptimal R-D performance) and dynamic segmentation scheme (high computational cost), and raises the following questions naturally:

- Can we achieve the oracle like R-D performance with computational ease?
- Can tree segmentation based algorithm *mimic* the oracle R-D performance for piecewise polynomial signals?

Therefore, our goal is to design a tree structured 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.

This chapter is organized as follows: In the next section, 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 2.4 and 2.5, we formally compute the R-D performance of

[1] 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 2.3.

these two coding schemes. Section 2.6 presents their computational complexity. Most importantly, we show that the prune-join tree algorithm, which jointly encodes similar neighbors, achieves optimal R-D performance (Theorem 2.2, Section 2.5) with computational ease (Section 2.6). In Section 2.7, we present simulation results which clearly demonstrate that the numerical results follow the theoretical results proven in the earlier sections. Finally, Section 2.8 offers concluding remarks.

2.2 Binary Tree Algorithms

In the previous section, we have seen that the wavelet based scheme performs suboptimally, whereas the dynamic segmentation scheme has high computational complexity. Thus, both schemes might have some limitations in practice. That is why, our target is to develop a compression algorithm based on the binary tree decomposition which achieves the oracle like R-D performance for piecewise polynomial signals (PPSs) with low computational cost. We first consider the prune binary tree algorithm. This algorithm is similar in spirit to the algorithm proposed in [45] 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 a rate-distortion framework with an MSE distortion metric. This algorithm can be described as follows:

Algorithm 2.1 ***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 $\widehat{J}$.[2]

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.[3]

Step 2: The Lagrangian cost based pruning

4. For the given operating slope $-\lambda$, the 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 [45]. The Lagrangian cost based

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

[3] This is best done in an orthogonal basis, that is, the Legendre polynomial basis.

pruning method is illustrated in Figure 2.2.

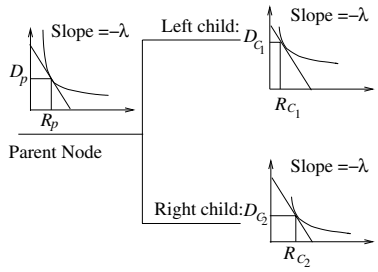


Figure 2.2: Lagrangian cost based pruning criterion for an operating slope $-\lambda$ for each parent node of the tree: Prune the children if $(D_{C_1} + D_{C_2}) + \lambda(R_{C_1} + R_{C_2}) \geq (D_p + \lambda R_p)$.

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 operates as follows [45]:

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 2.1, is depicted in Figure 2.3. 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.

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*

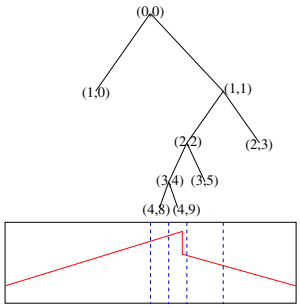


Figure 2.3: The prune binary tree segmentation.

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 2.1, 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 2.3 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 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.

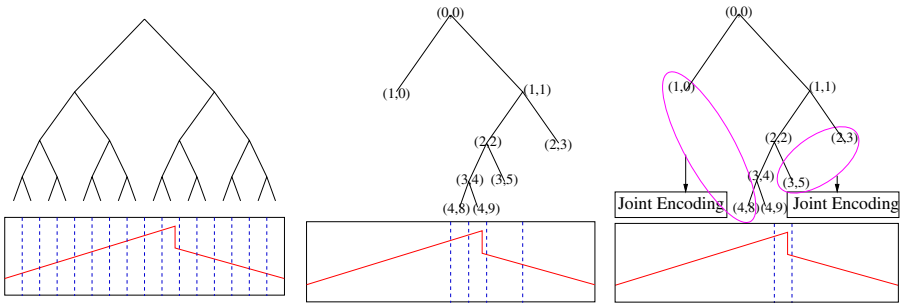
Now, if the current leaf is not the last leaf of the pruned tree, then the algorithm will restart the above described neighbor search and join operation for the next leaf of the pruned tree. Clearly, the neighbor joint coding variable is an indicator functional, which keeps track of the neighbor joining 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 2.2 ***The prune-join binary tree coding algorithm***

Step 1: Initialization
Following Steps 1 and 2 of Algorithm 2.1, 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. That means, the two neighbors 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, i.e., if $(D_{n_1} + \lambda R_{n_1}) + (D_{n_2} + \lambda R_{n_2}) \geq D_{n_{\text{Joint}}} + \lambda R_{n_{\text{Joint}}}$.
Step 3: Search for the desired R-D operating slope
Similar to Algorithm 2.1, iterate the process over λ until the bit budget constraint is met.

It is clearly visible in Figure 2.4(c) that the prune-join coding scheme is essentially coding a number of blocks equal to the number of segments like the oracle method. Therefore, we expect it to achieve the oracle like R-D performance for piecewise polynomial signals.[4]



(a) The full binary tree.

(b) The prune binary tree.

(c) The prune-join binary tree.

Figure 2.4: Comparative study of different tree segmentation algorithms.

[4]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.

2.3 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 [40] that the R-D behavior of the oracle based method decays as

$$D(R) \leq c_0 2^{-c_1 R}, \tag{2.1}$$

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

2.4 R-D Analysis of the Prune Binary Tree Coding Algorithm

This section presents the asymptotic R-D behavior of the prune tree coding algorithm for piecewise polynomial signals. We compute the worst case R-D upper-bound in the operational (algorithmic) sense.[5] This is to gain insight into the algorithm's performance at high rates and because it is difficult to compute the exact R-D function in a general setting such as ours.[6] First, we state a simple result in the form of Lemma 2.1 that the Lagrangian cost based pruning will prune the children nodes if the parent node has no singularity. Then we show that the prune tree algorithm encodes a number of leaves which grows linearly with respect to the decomposition depth J. This implies that several nodes with same parameters are coded separately (e.g., see Figure 2.4(b)). Finally, we prove that this independent coding of similar leaves results in a suboptimal R-D behavior given by Theorem 2.1.

Lemma 2.1 *At high bit rates, the Lagrangian cost* $(L = D(R) + \lambda R)$ *based tree pruning method prunes the children nodes if and only if the parent node does not contain a singularity.*

Proof: see Appendix 2.A. ◇

An intuitive explanation of the above result is as follows: If the parent node does not have a singularity, then its decomposition leads to the repetitive coding of the same polynomial, which is clearly suboptimal. And if the parent node

[5] Note that this differs from classic R-D theory, where exact (or tight upper/lower bounds) R-D behavior is computed for some exemplary processes.

[6] However, we also present the *expected* R-D lower-bound of the prune tree algorithm for piecewise constant signals with only one discontinuity in Appendix 2.C.

contains a singularity, then its decomposition provides better approximation, which will clearly improve the R-D performance at high rates.

Lemma 2.2 *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 nodes with singular points, as they cannot be well approximated by a polynomial (Lemma 2.1). 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

$$N_0 \quad \leq \quad 2S + (J-1)S = (J+1)S. \tag{2.2}$$

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

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

$$M_0 \quad \leq \quad 2JS + 1. \tag{2.3}$$

Theorem 2.1 *The prune binary tree coding algorithm, which employs the bottom-up R-D optimization using parent-children pruning, achieves the following asymptotic R-D behavior*

$$D_P(R) \leq c_2 \sqrt{R} 2^{-c_3\sqrt{R}}, \tag{2.4}$$

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 2.2) 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, so we approximate leaves with singularities by

the zero polynomial. 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 $D_l = A^2 (P+1)^2 T_l 2^{-\frac{2}{P+1}R_l}$ [40]. Since R-D optimal solution of exponentially decaying R-D functions results in equal distortion for each leaf [12], the coding algorithm will allocate the 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

$$\begin{aligned} \frac{\partial D_j}{\partial R_j} &= -\lambda, \forall j \geq 0 \\ \Rightarrow A^2 (P+1)^2 T 2^{-j} 2^{-\frac{2}{P+1}R_j} &= \frac{(P+1)\lambda}{2\ln 2}. \end{aligned} \tag{2.5}$$

Equation (2.5) 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

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

$$\Rightarrow R_j = R_k + \frac{P+1}{2}(k-j). \tag{2.6}$$

$$\Rightarrow R_{J+1} = R_J - \frac{P+1}{2} \leq R_J, \tag{2.7}$$

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.[7]

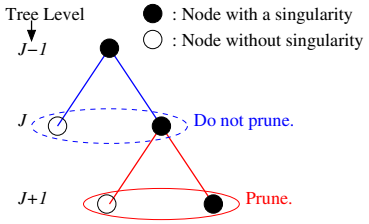


Figure 2.5: Figure shows the conditions to stop the pruning of a singularity containing node at the tree level J. That means, J becomes the tree-depth.

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 2.5): (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, $A^2T(P+1)^2 2^{-J} \leq A^2T(P+1)^2 2^{-(J+1)} + A^2T(P+1)^2 2^{-(J+1)} 2^{-\frac{2}{P+1}R_{J+1}} +$

[7] 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.

λR_{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, $A^2T(P+1)^2 2^{-(J-1)} \geq A^2T(P+1)^2 2^{-J} + A^2T(P+1)^2 2^{-J} 2^{-\frac{2}{P+1}R_J} + \lambda R_J$. These two conditions along with (2.5) and (2.7) mean that R_J must satisfy the following inequality

$$\frac{1}{2} \leq \quad 2^{-\frac{2}{P+1}R_J}\left(1+\frac{2\ln 2}{P+1}R_J\right) \quad \leq 1. \tag{2.8}$$

This is because (2.5) gives $\frac{(P+1)\lambda}{2\ln 2} = A^2T(P+1)^2 2^{-(J+1)} 2^{-\frac{2}{P+1}R_{J+1}} = A^2T(P+1)^2 2^{-J} 2^{-\frac{2}{P+1}R_J}$, and (2.7) provides $R_{J+1} \leq R_J$.

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[8]

$$\frac{P+1}{\ln 2} > \qquad R_J \qquad \geq 0, \text{ as } P \geq 0 \tag{2.9}$$

$$\Rightarrow \frac{1}{8} < 2^{-\frac{2}{\ln 2}} < \quad 2^{-\frac{2}{P+1}R_J} \quad \leq 1. \tag{2.10}$$

Multiplying the inequality (2.10) by $A^2T2^{-J}(P+1)^2$, we obtain

$$\frac{1}{8}A^2T2^{-J}(P+1)^2 < A^2T2^{-J}(P+1)^2 2^{-\frac{2}{P+1}R_J} \leq A^2T2^{-J}(P+1)^2. \tag{2.11}$$

The inequality (2.11) 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 (2.11) that, by choosing $R_J = 0$, we will obtain the worst case R-D performance. Thus, setting $R_J = 0$ and using (2.6), 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

$$\begin{aligned} D_P &\leq S\left(A^2T(P+1)^2 2^{-J}\right) + JS\left(A^2T\left(P+1\right)^2 2^{-J}\right) \\ \Rightarrow D_P &\leq A^2TS\left(P+1\right)^2\left(J+4\right)2^{-J}. \end{aligned} \tag{2.12}$$

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

$$R_{\text{Leaves}} = S\sum_{j=1}^{J}\frac{P+1}{2}(J-j) = S(P+1)\frac{J\left(J-1\right)}{4}. \tag{2.13}$$

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, (2.3) gives

[8]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.

$R_{\text{Tree}} \leq 2JS + 1$. The total bit 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

$$\begin{aligned} R &= R_{\text{Tree}} + R_{\text{Leaves}} \leq 2JS + 1 + \frac{(P+1)\,S}{4} J\,(J-1) \\ \Rightarrow R &\leq \frac{(P+1)\,S}{4}\,(J+4)^2\,; \text{ as } S > 0 \text{ and } J \text{ is large.} \end{aligned} \tag{2.14}$$

Combining (2.12) and (2.14) by eliminating J and noting that the right hand side of (2.12) is a decreasing function of J, whereas the right hand side of (2.14) 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. □

2.5 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 2.3 *At high bit rates, the Lagrangian cost based joining method jointly codes two neighboring nodes of the binary tree, if and only if the joint node has no singularity.*

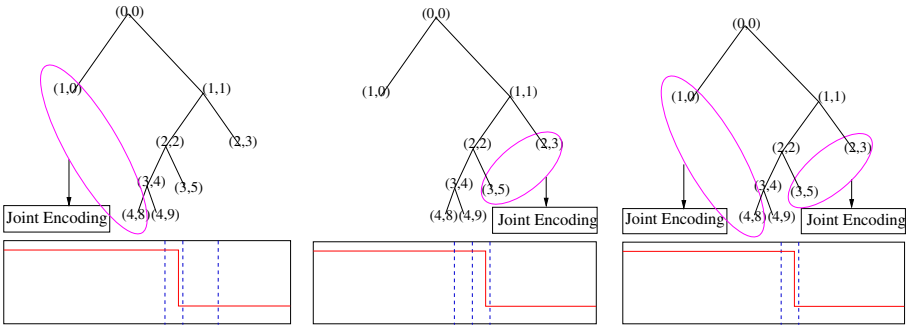
Proof: see Appendix 2.B. ◇

An intuitive explanation is as follows: If the joint node has no singularity, then neighboring nodes are essentially characterized by the same polynomial. Thus, joining of these nodes will improve the R-D performance by avoiding the repetitive coding of same information. On the other hand, if the joint node contains singularities, then separate coding of neighboring nodes provides better approximation and, thus, improves the R-D performance at high rates.

Lemma 2.4 *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: Lemma 2.3 guarantees that the neighboring leaves will be joined to improve the R-D performance if the joint block does not have a singularity. 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 2.6). Thus, the prune-join tree 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. ◇



(a) Joining of left leaves.

(b) Joining of right leaves.

(c) Complete joining.

Figure 2.6: Illustration of the prune-join binary tree joining.

Theorem 2.2 *The prune-join binary tree algorithm, which jointly encodes similar neighbors, achieves the oracle like exponentially decaying asymptotic R-D behavior*

$$D_{PJ}(R) \leq c_4 2^{-c_5 R}, \tag{2.15}$$

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 approximates singularity containing blocks with the zero 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 (2.3)), and $R_{\text{LeafJointCoding}} \leq (J+1)S$ (from (2.2)). 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

$$\begin{aligned} R &= R_{\text{Tree}} + R_{\text{LeafJointCoding}} + R_{\text{Leaves}} \\ R &\leq 2JS + 1 + (J+1)S + (S+1)\frac{(P+1)}{2}J \qquad (2.16) \\ \Rightarrow R &\leq \frac{(S(P+7)+(P+1))}{2}(J+1). \qquad (2.17) \end{aligned}$$

The net distortion bound is as follows

$$\begin{aligned} D_{PJ} &\leq SA^2T(P+1)^2 2^{-J} + (S+1)A^2T(P+1)^2 2^{-J} \\ &= (2S+1)(P+1)^2 A^2 T 2^{-J} = c_4 2^{-(J+1)} \\ \Rightarrow D_{PJ} &\leq c_4 2^{-\frac{2}{(S(P+7)+(P+1))}R}; \text{ from (2.17)}. \end{aligned}$$

Therefore, the prune-join tree algorithm achieves an 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 (2.1). One can notice in (2.16) 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 the similar neighbors.

2.6 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 samples, 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 polynomials*: 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:

$$\min_{\boldsymbol{p}} \|V_{L,P}\,\boldsymbol{p} - \boldsymbol{y}\|^2, \tag{2.18}$$

(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:

$$V_{L,P} = \begin{bmatrix} \phi_0(x_1) & \phi_1(x_1) & \phi_2(x_1) & \dots & \phi_P(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \phi_2(x_2) & \dots & \phi_P(x_2) \\ & & \dots & & \\ \phi_0(x_L) & \phi_1(x_L) & \phi_2(x_L) & \dots & \phi_P(x_L) \end{bmatrix}, \tag{2.19}$$

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).[9] 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 [41]. We can pre-compute and store the Legendre polynomial based Vandermonde matrix $V_{L,P}$ to use them 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 (2.18) is achieved efficiently by means of a QR factorization of $V_{L,P}$ with computational cost of $O(LP)$.[10] Since the average node-size is $O(\log N)$, the overall computational cost for computing the best polynomials for all nodes will be $O(N \log N)$.[11]

(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

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

[10] 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 [41, Chapter 3].

[11] For the complexity analysis, we have included the polynomial degree P in the complexity constant.

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* 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 $O(N \log R_Q)$.

3. *Iterative search algorithm for an optimal operating slope* calls the pruning algorithm for the chosen operating slope $-\lambda$. Our bisection search scheme obtains the optimal operating slope in $O(\log N)$ iterations [45]. 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. (2.2)) 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

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

2.7 Numerical Experiments

In this numerical experiment, we consider piecewise cubic polynomials with no more than $S = 16$ 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.

Figure 2.7 shows a particular realization of the piecewise cubic signal along with reconstructed signals by the prune and the prune-join binary tree schemes.

Figure 2.8 displays the segmentation and model allocation choices performed by the prune binary tree algorithm for different bit-rate constraints.[12] Similarly, in Figure 2.9, the segmentation and model allocation choices performed by the prune-join binary tree scheme are shown for different bit-rate constraints. In Figure 2.10, we compare R-D performance of the two proposed binary tree coding algorithms against their theoretical R-D behaviors. Figure 2.10 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 an exponentially decaying R-D behavior.

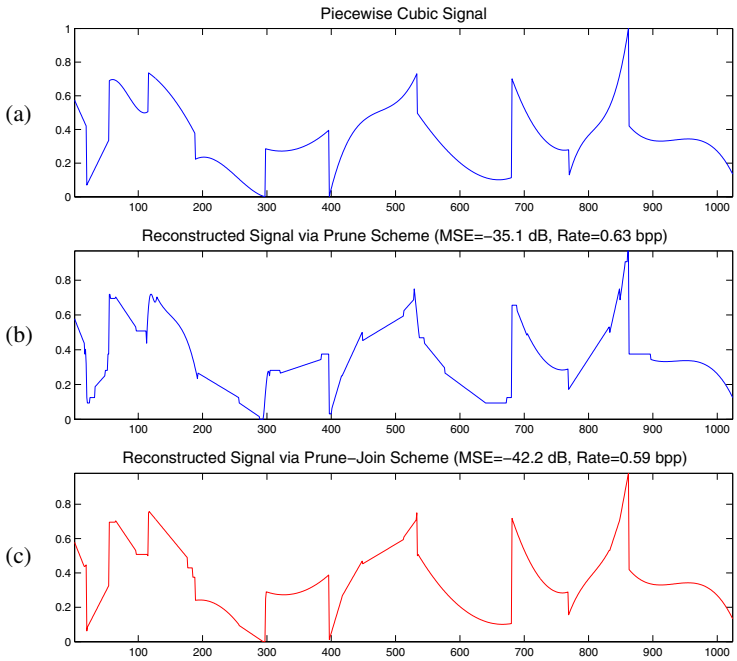
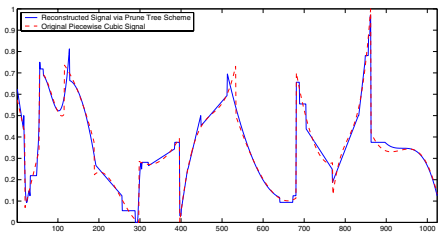


Figure 2.7: Original and reconstructed piecewise polynomial signals provided by the prune and prune-join binary tree algorithms. (a) Original piecewise cubic signal. (b) The prune tree algorithm: MSE= -35.1 dB, Bit-rate= 0.63 bps. (c) The prune-join tree algorithm: MSE= -42.20 dB, Bit-rate= 0.59 bps.

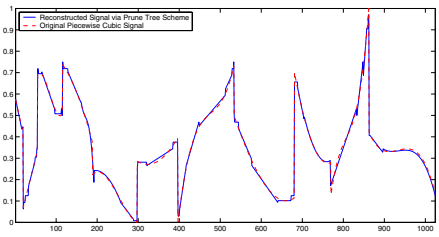
2.8 Conclusions

In this chapter, we have presented the prune and the prune-join binary tree compression algorithms and analyzed their asymptotic R-D performance, in an operational framework, for piecewise polynomial functions. We have proved that the prune tree coding algorithm performs suboptimally whereas the prune-join coding scheme achieves the oracle like exponentially decaying R-D behavior

[12] Bit-rate is given in bits per sample (bps).

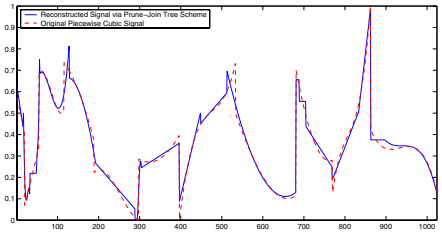


(a) MSE= −31.05 dB, Bit-rate= 0.4623 bps.

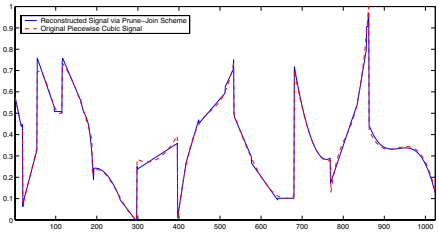


(b) MSE= −38.61 dB, Bit-rate= 0.7488 bps.

Figure 2.8: Approximations provided by the prune binary tree coding algorithm at different bit-rates.



(a) MSE= −30.98 dB, Bit-rate= 0.3889 bps.



(b) MSE= −38.39 dB, Bit-rate= 0.5675 bps.

Figure 2.9: Approximations provided by the prune-join binary tree coding algorithm at different bit-rates.

by using neighbor joint coding strategy. Moreover, we have also shown that the computational complexity of both schemes is $O(N \log N)$. That means, the proposed algorithms are computationally efficient and practical. Numerical results clearly indicate that the proposed prune-join compression algorithm represents an efficient way to model and approximate piecewise polynomial signals. Simulations results (Figure 2.10) also confirm that this algorithm achieves optimal performance if the input signal fits the model exactly. The theoretical as well as practical results seem to indicate that the prune-join binary tree coding scheme can also efficiently model and approximate more general class of piecewise smooth signals, which is the topic of investigation in the next chapter.

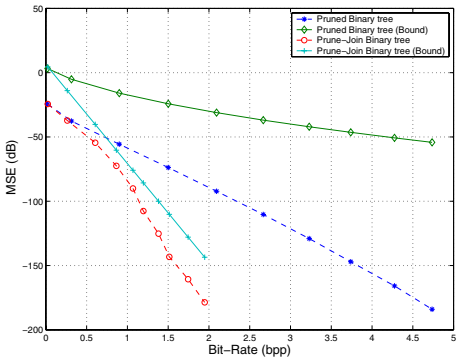


Figure 2.10: Theoretical (solid) and numerical (dotted) R-D curves for the prune and prune-join binary tree algorithms for piecewise polynomial signals.

Appendix 2.A Proof of Lemma 2.1: Parent Children Pruning

Assume that the parent node is representing a piecewise polynomial function $f(t)$, defined over the interval $[0, T]$, with polynomial pieces of maximum degree P. Let $p(t)$ be the best (least square error) polynomial approximation of $f(t)$. Hence, $p(t)$ minimizes the approximation distortion

$$D_\infty = \int_0^T (f(t) - p(t))^2 \, dt. \tag{2.20}$$

Since $p(t)$ minimizes the approximation distortion D_∞, the error function $(f(t) - p(t))$ must be orthogonal to any polynomial $q(t)$ of degree P. That means,

$$\int_0^T (f(t) - p(t)) \, q(t) \, dt = 0. \tag{2.21}$$

Clearly, $D_\infty = 0$ if $f(t)$ is simply a polynomial of maximum degree P, otherwise $D_\infty > 0$. Suppose that the polynomial $p(t)$ is further quantized to $\widehat{p}(t)$ according to the given bit budget R. As shown in [40],

$$\int_0^T (p(t) - \widehat{p}(t))^2 \, dt \leq A^2 T (P+1)^2 \, 2^{-\frac{2}{P+1}R}. \tag{2.22}$$

Therefore, the R-D function for the parent node can be written as follows

$$\begin{aligned} D(R) &= \int_0^T (f(t) - \widehat{p}(t))^2 = \int_0^T (f(t) - p(t) + p(t) - \widehat{p}(t))^2 \\ &= \int_0^T (f(t) - p(t))^2 + \int_0^T (p(t) - \widehat{p}(t))^2; \text{ due to orthogonality by (2.21)} \\ &\leq D_\infty + A^2 T (P+1)^2 \, 2^{-\frac{2}{P+1}R}; \text{ from (2.20) and (2.22).} \end{aligned} \tag{2.23}$$

Again, it is obvious that if the function $f(t)$ has no singularity, then $D_\infty = 0$. Since we are interested in the asymptotic R-D behavior, we will simply utilize the R-D bound given by (2.23) for further analysis. Thus, the asymptotic Lagrangian cost bound $L(\lambda)$ is

$$L(\lambda) = D_\infty + A^2 T (P+1)^2 2^{-\frac{2}{P+1}R} + \lambda R. \tag{2.24}$$

The minimization of the Lagrangian bound results into the following rate allocation

$$R = \frac{P+1}{2} \log_2 \left(\frac{2A^2 T (P+1) \ln 2}{\lambda} \right). \tag{2.25}$$

Combining (2.24) and (2.25), the minimum Lagrangian cost becomes

$$L(\lambda) = D_\infty + \frac{\lambda}{2} (P+1) \log_2 \left(\frac{2A^2 T (P+1) e \ln 2}{\lambda} \right). \tag{2.26}$$

Similarly, the minimum Lagrangian costs for the children nodes are as follows

$$L_1(\lambda) = D_{1\infty} + \frac{\lambda}{2} (P+1) \log_2 \left(\frac{2A^2 T_1 (P+1) e \ln 2}{\lambda} \right) \tag{2.27}$$

$$L_2(\lambda) = D_{2\infty} + \frac{\lambda}{2} (P+1) \log_2 \left(\frac{2A^2 T_2 (P+1) e \ln 2}{\lambda} \right), \tag{2.28}$$

where $D_{1\infty}$ and $D_{2\infty}$ represent the polynomial approximation error for the first and second segment at infinite rate, respectively and $T_1 = T_2 = \frac{T}{2}$. Thus, the difference in the Lagrangian cost of the parent and that of its children can be written as follows

$$\begin{aligned} \Delta L(\lambda) &= L(\lambda) - L_1(\lambda) - L_2(\lambda) \\ &= (D_\infty - D_{1\infty} - D_{2\infty}) + \frac{\lambda}{2} (P+1) \log_2 \left(\frac{\lambda T}{2A^2 T_1 T_2 (P+1) e \ln 2} \right). \end{aligned}$$

If the signal has no singularity, then $D_\infty = D_{1\infty} = D_{2\infty} = 0$. Therefore, the difference in the Lagrangian costs becomes

$$\Delta L(\lambda) = \frac{\lambda}{2} (P+1) \log_2 \left(\frac{\lambda T}{2A^2 T_1 T_2 (P+1) e \ln 2} \right)$$

$$\Delta L(\lambda) < 0; \text{ for } \lambda < \lambda_0 = \frac{2}{T} A^2 T_1 T_2 (P+1) e \ln 2. \tag{2.29}$$

Clearly , at high rates, $R \geq 2$, so $\lambda \leq \frac{1}{2} A^2 T < \lambda_0$.[13] Thus, (2.29) ensures that the Lagrangian cost of the parent is smaller than the sum of the Lagrangian costs of its children if the parent has no singularity. Hence, the children are pruned.

[13] For a rate of zero bits, the resulting distortion D_0 is equal to the energy of the entire signal. That means, $D_0 = A^2 T$. Let λ be such that $(R^*(\lambda), D^*(\lambda))$ is optimum with $R^*(\lambda) \geq 2$ bits. Due to the R-D optimization, for all (R, D) pairs, $D + \lambda R \geq D^*(\lambda) + \lambda R^*(\lambda)$. Since this relation is also true for $(0, D_0)$, the following holds : $\lambda \leq \frac{D_0 - D^*(\lambda)}{R^*(\lambda)} \leq \frac{1}{2} A^2 T < \lambda_0$.

Similarly, if the signal has a singularity, then one can easily show that $D_\infty - D_{1\infty} - D_{2\infty} > 0$.[14] Therefore, the difference in the Lagrangian costs becomes

$$\begin{aligned} \Delta L(\lambda) &= (D_\infty - D_{1\infty} - D_{2\infty}) + \frac{\lambda}{2}(P+1)\log_2\left(\frac{\lambda T}{2A^2 T_1 T_2 (P+1) e \ln 2}\right) \\ &= (D_\infty - D_{1\infty} - D_{2\infty})\,; \text{ as at high rates, } \lambda \to 0 \text{ and } \lim_{\lambda\to 0} \lambda \log \lambda \to 0 \\ \Rightarrow \Delta L(\lambda) &> 0. \end{aligned} \tag{2.30}$$

Hence, the Lagrangian cost of the parent is greater than the sum of the Lagrangian costs of its children and the algorithm decides to split the parent. Therefore, (2.29) and (2.30) clearly show that the Lagrangian cost of the parent is smaller than the sum of the Lagrangian costs of the children if and only if the parent has no singularity. This implies that, at high rates, the children are pruned if and only if the parent contains no singularity. ◇

Appendix 2.B Proof of Lemma 2.3: Neighbor Joining

In essence, the proof of Lemma 2.3 closely follows the logic outlined for the proof of Lemma 2.1. Suppose that we have two neighboring blocks of size T_1 and T_2, then the joint block (node) will be of size $T = (T_1 + T_2)$. This can be interpreted as the segmentation of the joint block into the two segments of size T_1 and T_2. Then (2.29) and (2.30) clearly show that the Lagrangian cost of the joint node will be smaller than the sum of the Lagrangian costs of the neighboring blocks if and only if the joint block has no singularity. Therefore, at high rates, the neighboring nodes will be merged if and only if the joint block contains no singularity. ◇

[14] It is also intuitive as the segmentation provides the finer representation for the complete signal.

Appendix 2.C R-D Lower-Bound of the Prune Binary Tree Coding Algorithm

In this section, we consider signals which are composed of two constant functions separated by a singularity. That means, $S = 1$. The resulting signal is simply a step function defined over the interval $[0, T]$ (see the example in Figure 2.11). Assume that the singularity location t_0 is uniformly distributed over the support $[0, T]$ and the values of the function left and right of the discontinuity are uniformly distributed over $[-A, A]$. Up to now, we have computed the absolute R-D upper bounds. But, in the following we derive the *expected* R-D lower-bound for the proposed prune tree algorithm.

In the single singularity case, at every tree level there is exactly one node to be divided further to improve the R-D performance at high rates (see Figure 2.11). Thus, the inequalities in (2.2) and (2.3) become equalities and can be written as follows.

$$N_0 = J + 1 \tag{2.31}$$

$$M_0 = 2J + 1. \tag{2.32}$$

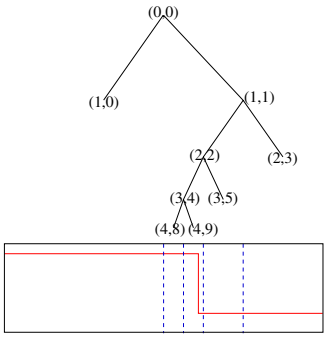


Figure 2.11: The prune binary tree segmentation of a piecewise constant signal with one singularity.

Moreover, every tree level $j > 0$ has exactly one leaf without singularity of size $T_j = T2^{-j}$ and the singularity containing leaf is at the tree-depth J. The distortion of the singularity containing leaf is D_{Sing} and $0 \leq D_{\text{Sing}} \leq A^2T2^{-J}$. The remaining J leaves can be simply represented by a constant function. Under the high resolution hypothesis for a r-bit quantizer, the quantization error for a constant coefficient becomes a uniform random variable over an interval $[-A2^{-r}, A2^{-r}]$. Therefore, the *expected* squared error for each quantized constant coefficient is $\frac{1}{3}A^2 2^{-2r}$. If the algorithm allocates r_j bits to a leaf without singularity at level j, the corresponding *expected* distortion D_j for a leaf j of size $T2^{-j}$ is given by

$$D_j = \frac{1}{3}A^2 2^{-2r_j} T2^{-j}. \tag{2.33}$$

Therefore, the overall *expected* distortion for the prune tree algorithm can be written as follows

$$D_P = \sum_{j=1}^{J} D_j + D_{\text{Sing}} \tag{2.34}$$

$$D_P \geq \sum_{j=1}^{J} \frac{1}{3} A^2 T 2^{-j} 2^{-2r_j}; \text{ as } D_{\text{Sing}} \geq 0. \tag{2.35}$$

Since the cost of coding the pruned binary tree is $R_{\text{Tree}} = M_0 = 2J + 1$, the net rate becomes

$$\begin{aligned} R &= \sum_{j=1}^{J} r_j + R_{\text{Tree}} \\ R &= \sum_{j=1}^{J} r_j + 2J + 1. \end{aligned} \tag{2.36}$$

Our goal is to find $r_1, r_2, \ldots, r_J$ and J such that the following Lagrangian cost functional $L(\lambda)$ is minimized[15]

$$L(\lambda) = \sum_{j=1}^{J} \frac{1}{3} A^2 T 2^{-j} 2^{-2r_j} + \lambda \left(\sum_{j=1}^{J} r_j + 2J + 1 \right). \tag{2.37}$$

That means,

$$\begin{aligned} \frac{\partial L}{\partial r_j} &= 0, \quad \forall j = 1, \cdots, J \\ \Rightarrow \frac{1}{3} A^2 T 2^{-j} 2^{-2r_j} &= \frac{\lambda}{2 \ln 2}; \quad \text{equal distortion constraint} \qquad (2.38) \\ \Rightarrow r_j &= r_k + \frac{k-j}{2}; \text{ similar to (2.6)} \\ \Rightarrow r_j &= r_1 + \frac{1-j}{2}. \qquad (2.39) \end{aligned}$$

By combining (2.37), (2.38) and (2.39), the Lagrangian cost functional can be rewritten as follows

$$L(\lambda) = J \frac{\lambda}{2 \ln 2} + \lambda \left(J r_1 - \frac{J(J-1)}{4} + 2J + 1 \right). \tag{2.40}$$

And

$$\begin{aligned} \frac{\partial L}{\partial J} &= 0 \\ \Rightarrow \frac{\lambda}{2 \ln 2} + \lambda \left(r_1 - \frac{2J-1}{4} + 2 \right) &= 0 \\ \Rightarrow r_1 &= \frac{2J-1}{4} - 2 - \frac{1}{2 \ln 2}. \end{aligned} \tag{2.41}$$

[15] This R-D bound based Lagrangian cost functional is obtained by combining the right hand sides of (2.35) and (2.36).

Now, by combining (2.36) and (2.39), the total rate can be rewritten as follows

$$\begin{aligned} R &= Jr_1 - \frac{J(J-1)}{4} + 2J + 1 \\ &= J\left(\frac{2J-1}{4} - 2 - \frac{1}{2\ln 2}\right) - \frac{J(J-1)}{4} + 2J + 1; \text{ from (2.41)} \\ \Rightarrow R &\geq \frac{1}{4}(J-2)^2. \end{aligned} \tag{2.42}$$

From (2.38), it is also clear that

$$\frac{1}{3}A^2T2^{-1}2^{-2r_1} = \frac{\lambda}{2\ln 2}. \tag{2.43}$$

Combining (2.41) and (2.43), we obtain

$$\frac{16e}{3}A^2T2^{-J} = \frac{\lambda}{2\ln 2}. \tag{2.44}$$

By combining (2.35), (2.38) and (2.44), we obtain the following R-D lower-bound

$$\begin{aligned} D_P &\geq J\frac{16e}{3}A^2T2^{-J} \qquad (2.45) \\ D_P &\geq \frac{8e}{3}A^2T\left(\sqrt{R}+1\right)2^{-2\sqrt{R}}; \text{ combining (2.42) and (2.45).} \\ \Rightarrow D_P &\geq \frac{8e}{3}A^2T\sqrt{R}2^{-2\sqrt{R}} \end{aligned}$$

The above result is summarized in the following theorem.

Theorem 2.3 *The prune binary tree coding algorithm, which employs the bottom-up R-D optimization using parent-children pruning, achieves an asymptotic R-D behavior which is lower bounded (in expectation) as follows*

$$D_P(R) \geq c_2'\sqrt{R}2^{-c_3'\sqrt{R}}, \tag{2.46}$$

where $c_2' = \frac{8e}{3}A^2T$ *and* $c_3' = 2$*, for piecewise constant signals with only one singularity.*

Theorem 2.3 also indicates that, on average, the prune binary tree algorithm will necessarily perform worse than (2.46) for more general piecewise polynomial signals with multiple singularities. Therefore, on average, the prune binary tree coding algorithm performs suboptimally.

Since the prune-join tree algorithm already achieves the oracle like exponentially decaying R-D behavior (Theorem 2.2) , computation of its R-D lower-bound will not provide any interesting information. That is why the lower-bound analysis is not presented for the prune-join tree algorithm.

Chapter 3

Binary Tree Segmentation Algorithms and 1-D Piecewise Smooth Signals

3.1 Introduction

It is intuitive that many signals encountered in practice can be closely modeled as piecewise smooth. Thus, if a coding scheme can accurately represent piecewise smooth signals, it might have an important impact on signal processing applications like approximation and compression of real life signals. That is why, in this chapter, we study whether the proposed binary tree segmentation algorithms can accurately model piecewise smooth signals. In the chapter, our main focus is on the tree segmentation based polynomial approximation of 1-D piecewise smooth signals. In the next section, we analyze the R-D performance of the binary tree algorithms for 1-D smooth signals. We then extend this R-D analysis to the more general piecewise smooth signals in Section 3.3. In particular, we show that both the prune and prune-join tree schemes are able to achieve the oracle like R-D behavior. In Section 3.4, we present experimental results which are also consistent with the theoretical analysis. Finally, Section 3.5 offers concluding remarks.

3.2 R-D Analysis for 1-D Smooth Functions

Our interest is in piecewise smooth signals, which are composed of regular pieces. The regularity of a function is generally measured by the Hölder exponent [32, 33].

Definition 3.1

- A function $f(t)$ is pointwise Hölder $\alpha \geq 0$ at t_0, if there exist a $K > 0$, and a polynomial $p_{t_0}(t)$ of degree $P = \lfloor \alpha \rfloor$ such that[1]

$$\forall t \in \mathbb{R}, |f(t) - p_{t_0}(t)| \leq K|t - t_0|^{\alpha}. \tag{3.1}$$

- A function $f(t)$ is uniformly Hölder α over $[a, b]$ if it satisfies (3.1) for all $t_0 \in [a, b]$, with a constant K that is independent of t_0.

- The Hölder regularity of $f(t)$ at t_0 or over $[a, b]$ is the supremum of the α such that $f(t)$ is Hölder α.

Moreover, for a uniformly Hölder α smooth function $f(t)$, the polynomial $p_{t_0}(t)$, in (3.1), is the P^{th} order Taylor series expansion of $f(t)$ around the point t_0. Since the regularity of a function guarantees that the function can be well approximated by an appropriate polynomial, our aim is to investigate whether the binary tree segmentation based coding algorithm, which utilizes this fact, can achieve the correct R-D performance.

3.2.1 R-D Performance of the Oracle Method

Consider a continuous-time smooth signal $f(t)$, which is uniformly Hölder α regular according to Definition 3.1. For such a signal, it has been shown in [9] that the best asymptotic R-D upper-bound is as follows

$$D(R) \leq CR^{-2\alpha}, \tag{3.2}$$

for some constant $C > 0$.

3.2.2 R-D Analysis of the Binary Tree Algorithms

Theorem 3.1 *For 1-D smooth functions, which are uniformly Hölder α over $[0, T]$, the full binary tree decomposition algorithm, which codes every binary tree node at the tree depth by a polynomial approximation, achieves the following asymptotic R-D behavior*

$$D_{Full}(R) \leq d_0 \left(\frac{\log R}{R} \right)^{2\alpha}, \tag{3.3}$$

where d_0 is a positive constant which depends on the smoothness, magnitude and region of support of the function.

Proof: Consider a 1-D function $f(t)$, which is uniformly Hölder α smooth and defined over the region $[0, T]$. The binary tree decomposition leads to the dyadic intervals of size $T2^{-J}$ at the depth J. These intervals can be represented by

[1] To be more precise, α and P are such that $P < \alpha \leq P + 1$.

$\mathcal{R}_i = [i,(i+1)]T2^{-J}; 0 \le i < 2^J$. The binary tree algorithm approximates the function in each segment $\mathcal{R}_i$ by its best (least square error) polynomial approximation $p_i(t)$ of degree $P = \lfloor \alpha \rfloor$. Since $p_i(t)$ is the best P^{th} order polynomial approximation of the function $f(t)$ on the region $\mathcal{R}_i$, $f(t) - p_i(t)$ will be orthogonal to any P^{th} order polynomial $q(t)$ on the region $\mathcal{R}_i$. That means,

$$\int_{\mathcal{R}_i} (f(t) - p_i(t))\, q(t) \quad = \quad 0. \tag{3.4}$$

Due to (3.1), the polynomial approximation squared error for the region $\mathcal{R}_i$ can be bounded as follows

$$\begin{aligned} \int_{\mathcal{R}_i} (f(t) - p_i(t))^2 &\le \int_{\mathcal{R}_i} K_i^2 |t - x_i|^{2\alpha} \\ &\le K_i^2 \int_{\mathcal{R}_i} (T2^{-J})^{2\alpha} \\ &\le K_{\max}^2 T^{2\alpha} 2^{-2\alpha J} T 2^{-J} \\ &= K_0 2^{-2\alpha J} T 2^{-J}; \quad K_0 = K_{\max}^2 T^{2\alpha}, \end{aligned} \tag{3.5}$$

where x_i is the origin coordinate of the tree node associated with region $\mathcal{R}_i$ and $K_{\max} = \max_{0 \le i < 2^J} K_i$.

The binary tree algorithm codes the polynomial $p_i(t)$ by quantizing its $(P+1)$ coefficients. Assume that the algorithm allocates r_i bits to the polynomial $p_i(t)$. Suppose that the polynomials $p_i(t), i = 0, \ldots, 2^J - 1$, are bounded in magnitude by some constant A. Thus, the polynomial quantization distortion d_i for the region $\mathcal{R}_i$ due to the coefficient quantization can be bounded as follows [40]

$$\begin{aligned} d_i &= \int_{\mathcal{R}_i} (p_i(t) - \widehat{p}_i(t))^2 \\ &\le A_0^2 T 2^{-J} 2^{-\frac{2}{P+1} r_i}; \quad A_0 = A(P+1). \end{aligned} \tag{3.6}$$

Hence, the coding distortion D_i for the region $\mathcal{R}_i$ is as follows

$$\begin{aligned} D_i &= \int_{\mathcal{R}_i} (f(t) - \widehat{p}_i(t))^2 = \int_{\mathcal{R}_i} (f(t) - p_i(t) + p_i(t) - \widehat{p}_i(t))^2 \\ &= \int_{\mathcal{R}_i} (f(t) - p_i(t))^2 + \int_{\mathcal{R}_i} (p_i(t) - \widehat{p}_i(t))^2, \text{ due to (3.4)} \\ &\le K_0 2^{-2\alpha J} T 2^{-J} + A_0^2 T 2^{-J} 2^{-\frac{2}{P+1} r_i}, \text{ from (3.5) and (3.6).} \end{aligned} \tag{3.7}$$

The binary tree scheme provides the piecewise polynomial approximation $p(t) = \sum_{i=0}^{2^J-1} p_i 1_{\mathcal{R}_i}(t)$, which is further quantized to $\widehat{p}(t) = \sum_{i=0}^{2^J-1} \widehat{p}_i(t) 1_{\mathcal{R}_i}(t)$. There-

fore, the overall coding distortion can be written as follows

$$\begin{aligned} D &= \int_{[0,T]} \left(f(t) - \widehat{p}(t)\right)^2 \\ &= \sum_{i=0}^{2^J-1} \int_{\mathcal{R}_i} \left(f(t) - \widehat{p}_i(t)\right)^2 \\ &\leq \sum_{i=0}^{2^J-1} \left[K_0 2^{-2\alpha J} T 2^{-J} + A_0^2 T 2^{-J} 2^{-\frac{2}{P+1} r_i}\right], \text{ from (3.7)} \\ &= K_0 T 2^{-2\alpha J} + \sum_{i=0}^{2^J-1} A_0^2 T 2^{-J} 2^{-\frac{2}{P+1} r_i}. \end{aligned} \tag{3.8}$$

The structure of (3.8) ensures that R-D optimization results in the equal distortion for all the leaves.[2] That means, all the leaves will be allocated the same rate, i.e. $r_i = r, \forall i = 0, \ldots, 2^J - 1$. Moreover, we can notice from (3.7) that the distortion of a leaf is the sum of two terms and the best R-D performance for a leaf will be achieved when both terms are of the same order.[3] Hence, for the given bit-budget R, the algorithm selects J and r such that the distortion of each leaf is of the order $O(2^{-2\alpha J - J})$. Thus, the scheme allocates $r = (P+1)\alpha J$ bits to each polynomial associated with a leaf. Therefore, the net distortion can be expressed as follows

$$\begin{aligned} D &\leq (K_0 + A_0^2) T 2^{-2\alpha J} \\ &= K_1 2^{-2\alpha J}; \quad K_1 = (K_0 + A_0^2) T \\ &\leq K_1 \left(1 + \frac{\log J}{J}\right)^{2\alpha} 2^{-2\alpha J}, \text{as at high rates } J > 1. \end{aligned} \tag{3.9}$$

Since the full binary tree decomposition algorithm does not need to code the tree structure, the global bit budget is simply equal to the total bits required for quantizing the polynomial coefficients associated with 2^J binary tree leaves at the tree depth J. Hence, the net rate is as follows

$$\begin{aligned} R &= 2^J r = 2^J (P+1)\alpha J \\ R &= K_2 J 2^J; \quad K_2 = (P+1)\alpha, \\ \log_2\left(\frac{R}{K_2}\right) &= \log_2 J + J. \end{aligned} \tag{3.10}$$

[2] This result can be derived using the usual reverse water filling technique. That is, $\frac{\partial D}{\partial r_i} =$ constant.

[3] Because if both terms are not of the same order, then one of the terms will dominate the overall distortion and the R-D performance can be improved by reselecting J and r.

Combining (3.9) and (3.10) leads to the following asymptotic R-D behavior

$$\begin{aligned} D &\leq K_1 \left(\frac{\log_2 \left(\frac{R}{K_2} \right)}{\frac{R}{K_2}} \right)^{2\alpha} \\ &\leq K_1 K_2^{2\alpha} \left(\frac{\left(K_2 + \frac{1}{K_2} \right) \log R}{R} \right)^{2\alpha}, \text{ as at high rates } R > 2 \\ D &\leq d_0 \left(\frac{\log R}{R} \right)^{2\alpha}, \end{aligned} \tag{3.11}$$

where $d_0 = K_1 \left(K_2^2 + 1 \right)^{2\alpha} = \left(K_{\max}^2 T^{2\alpha} + A_0^2 \right) T \left((P+1)^2 \alpha^2 + 1 \right)^{2\alpha}$. □

Corollary 3.1 *The prune binary tree coding algorithm achieves the following asymptotic R-D behavior*

$$D_{Prune}(R) \leq d_0 \left(\frac{\log R}{R} \right)^{2\alpha}, \tag{3.12}$$

for 1-D smooth functions, which are uniformly Hölder α over $[0, T]$.

Proof: Since the prune binary tree algorithm results into the full binary tree algorithm, when no pruning is performed, the prune binary tree scheme will perform at least as well as the full tree scheme. Thus, the prune tree scheme will also achieve the R-D behavior given by (3.12). ◇

Corollary 3.2 *The prune-join binary tree algorithm achieves the following asymptotic R-D behavior*

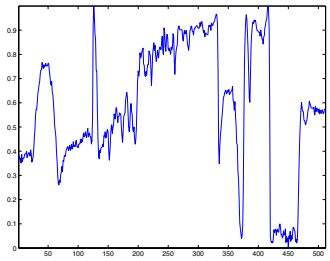
$$D_{Prune\text{-}Join} \leq d_0 \left(\frac{\log R}{R} \right)^{2\alpha}, \tag{3.13}$$

for 1-D smooth functions, which are uniformly Hölder α over $[0, T]$.

Proof: Since the prune-join binary tree algorithm leads to the full binary tree algorithm, when pruning and neighbor joining are not performed, the prune-join tree scheme will perform at least as well as the full tree scheme. Thus, the prune-join tree scheme will also achieve the R-D behavior shown by (3.13). ◇

Therefore, for uniformly Hölder α smooth 1-D functions, the full, the prune and the prune-join schemes achieve the similar asymptotic R-D behavior $\left(d_0 \left(\frac{\log R}{R} \right)^{2\alpha} \right)$. However, note that the given R-D bound is a weak upper-bound for both the prune and the prune-join coding schemes.

(a) Lena image.



(b) A line of the Lena image.

Figure 3.1: Original lena image and its scanned line.

3.3 R-D Analysis for 1-D Piecewise Smooth Functions

We can define a piecewise smooth function $f(t)$, $t \in [0, T]$ with $S+1$ pieces, as follows

$$f(t) = \sum_{i=0}^{S} f_i(t) \mathbf{1}_{[t_i, t_{i+1})}(t), \tag{3.14}$$

where $t_0 = 0$, $t_{S+1} = T$ and $f_i(t)$ is uniformly Hölder α over $[0, T]$. These type of signals are interesting, because many signals encountered in practice can be modeled as piecewise smooth. For example, in Figure 3.1(b) we show a line of the Lena image (Figure 3.1(a)), as one can see this signal is very close to a piecewise smooth signal.

The following R-D analysis rely on a key theorem proven by Dragotti *et al.* [21], which states that a 1-D piecewise smooth function $f(t)$, whose pieces are Hölder α-smooth, can be expressed as a sum of a 1-D piecewise polynomial $p(t)$ with pieces of degree no more than $P = \lfloor \alpha \rfloor$ and a residual function $r(t)$ which is Hölder α-smooth. We can formally state the theorem as follows:

Theorem 3.2 ***(Dragotti and Vetterli, [21])*** *Given is a piecewise smooth signal $f(t)$ defined as in Eq. (3.14), that is, with pieces of Hölder regularity α. Then, there exists a piecewise polynomial signal $p(t)$ with pieces of maximum degree $P = \lfloor \alpha \rfloor$ such that the difference signal $r_\alpha(t) = f(t) - p(t)$ is uniformly Hölder α over $[0, T]$.*

3.3.1 R-D Performance of the Oracle Method

Consider a continuous-time piecewise smooth signal $f(t)$ defined by (3.14). For such a signal, it was shown in [20] that the best asymptotic R-D upper bound

is as follows

$$D(R) \leq C R_s^{-2\alpha} + d_1 2^{-d_2 R_p}, \tag{3.15}$$

where $R = R_s + R_p$; R_s and R_p are bitrates used to code the smooth residual and the piecewise polynomial signal, respectively.

3.3.2 R-D Analysis of the Binary Tree Algorithms

Theorem 3.3 *For 1-D piecewise smooth functions with uniformly Hölder α smooth pieces, the prune binary tree algorithm, which employs the parent-children pruning, achieves the following asymptotic R-D behavior*

$$D_{Prune}(R) \leq d_3 \left(\frac{\log R}{R}\right)^{2\alpha}, \tag{3.16}$$

where d_3 is a positive constant which depends on the characteristics, like number of smooth pieces and their smoothness, of piecewise smooth functions.

Proof: The following asymptotic R-D analysis utilizes the insights gained from the analysis of piecewise polynomial signals (Section 2.4) and that of smooth functions (Section 3.2.2). From Section 2.4, we learn that the R-D optimization performs finer segmentation in the singularity containing regions[4] and also selects a bit allocation strategy to ensure that all the tree leaves have the distortion of the same order. On the other hand, Section 3.2.2 tells us that a smooth function can be well represented by segmentation and corresponding piecewise polynomial approximation.

Assume that the 1-D piecewise smooth function $f(t)$, as defined in (3.14), is decomposed using binary tree algorithm up to the tree depth J. Since the signal contains only S singularities, each tree level j can have at most S singularity containing nodes and the remaining $2^j - S$ nodes can be simply represented by smooth functions. Clearly, at high rates, for achieving better R-D performance the tree pruning scheme will decompose the nodes with singular points deeper compared to nodes without singularities (smooth nodes). Suppose that the prune tree algorithm splits smooth nodes only up to the tree level J_1 and beyond this level only singularity containing nodes are further splitted.[5] Since the prune tree scheme can split at most S singularity containing nodes beyond the tree level J_1, every level beyond J_1, except $j = J$, will generate at most S leaves. The level J will have $2S$ leaves, while the level J_1 will have $2^{J_1} - S$ leaves. Thus, the total number N_0 of leaves can be bounded as follows

$$N_0 \quad \leq \quad \left(2^{J_1} - S\right) + 2S + (J - 1 - J_1)\, S = 2^{J_1} + (J - J_1)S. \tag{3.17}$$

Moreover, it can also be noted that in the pruned tree, every tree level $j > J_1$ can have at most $2S$ nodes. Hence, the total number M_0 of nodes in the pruned

[4]Because polynomials cannot model segments with singularities.

[5]This means that there is no leaf up to the tree level $J_1 - 1$.

tree can be given as follows

$$M_0 \leq 2 \cdot 2^{J_1} + 2S\,(J - J_1)\,. \tag{3.18}$$

Clearly, in the worst case scenario, all the singularity containing leaves 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}.[6] We can notice from (3.7) that the distortion of a smooth leaf l at tree level j is bounded by $K_0 2^{-2\alpha j} T2^{-j} + A_0^2 T2^{-j}2^{-\frac{2}{P+1}r_l}$, where r_l is the rate allocated to the smooth leaf. Again, R-D optimization requires that all the tree leaves must have the distortion of the same order. Thus, the algorithm selects $J = (2\alpha+1)J_1$ and allocates $(P+1)\alpha J_1$ bits to each polynomial associated with a smooth leaf. This ensures that all the leaves have distortions of similar order $O\left(2^{-(2\alpha+1)J_1}\right)$. Therefore, the total distortion can be bounded as follows

$$\begin{aligned} D &\leq N_0\left(K_0 + A_0^2\right) T2^{-J_1}2^{-2\alpha J_1} \\ &\leq \left(2^{J_1} + 2\alpha J_1 S\right)\left(K_0 + A_0^2\right) T2^{-J_1}2^{-2\alpha J_1}, \text{ from (3.17)} \\ &\leq (2\alpha S + 1)2^{J_1}\left(K_0 + A_0^2\right) T2^{-J_1}2^{-2\alpha J_1} \\ &= K_1 2^{-2\alpha J_1}; \qquad K_1 = (2\alpha S + 1)\left(K_0 + A_0^2\right) T. \end{aligned} \tag{3.19}$$

The total bitrate is the sum of the costs of coding the tree structure and the quantized model parameters of the smooth leaves. The bitrate R_{Tree} required to code the tree structure is equal to the total number M_0 of nodes in the pruned tree. Thus, (3.18) gives $R_{\text{Tree}} \leq 2 \cdot 2^{J_1} + 4\alpha S J_1 \leq 2(2\alpha S + 1)2^{J_1}$. Since the algorithm allocates $(P+1)\alpha J_1$ bits to leaves without singularities and zero bits to leaves with singularities, the bitrate R_{Leaves} needed for coding the leaves is bounded as follows: $R_{\text{Leaves}} \leq N_0(P+1)\alpha J_1 \leq (2\alpha S+1)(P+1)\alpha J_1 2^{J_1}$. Hence, the total bit rate can be expressed as follows

$$\begin{aligned} R &= R_{\text{Tree}} + R_{\text{Leaves}} \\ &\leq 2(2\alpha S + 1)2^{J_1} + (2\alpha S + 1)(P + 1)\alpha J_1 2^{J_1} \\ &\leq (2\alpha S + 1)((P + 1)\alpha + 2)J_1 2^{J_1}; \text{ as at high rates } J_1 \geq 1 \\ \Rightarrow R &\leq K_2 J_1 2^{J_1}; \qquad K_2 = (2\alpha S + 1)((P + 1)\alpha + 2). \end{aligned} \tag{3.20}$$

Combining (3.19) and (3.20) provides the following asymptotic R-D behavior

$$D \leq d_3 \left(\frac{\log R}{R}\right)^{2\alpha}, \tag{3.21}$$

where $d_0 = K_1 K_2^{2\alpha} = (2\alpha S + 1)\left(K_0 + A_0^2\right) T(2\alpha S + 1)^{2\alpha}((P + 1)\alpha + 2)^{2\alpha}$. Therefore, the prune binary tree algorithm achieves the announced R-D behavior. □

[6] Recall that the algorithm approximates leaves with singularities by the zero polynomial.

Corollary 3.3 *For 1-D piecewise smooth functions with uniformly Hölder α smooth pieces, the prune-join binary tree algorithm, which performs the joint coding of similar neighbors, achieves the following asymptotic R-D behavior*

$$D_{Prune\text{-}Join}(R) \leq d_3 \left(\frac{\log R}{R} \right)^{2\alpha}. \tag{3.22}$$

Proof: Since the prune-join binary tree algorithm results into the prune binary tree algorithm, when no neighbor joining is performed, the prune-join tree scheme will perform atleast as good as the prune tree scheme. Hence, the prune-join tree scheme will also achieve the asymptotic R-D behavior indicated by (3.22). ◇

Therefore, both the prune and prune-join binary tree algorithms achieve the polynomially decaying asymptotic R-D behavior for piecewise smooth functions.

3.4 Simulation Results

In this experimental setup, we consider piecewise smooth signals with no more than $S = 16$ singularities and each smooth piece is twice differentiable.

Figure 3.2 shows a particular realization of the piecewise smooth signal. Figure 3.3 displays the segmentation and model allocation choices performed by the prune and prune-join binary tree algorithms for the given bit-rate constraints. Similarly, in Figure 3.4, the residual signals obtained by the prune and prune-join binary tree scheme are shown. As predicted by the theory, the residual signals are smooth. In Figure 3.5, we compare R-D performance of the two proposed binary tree coding algorithms against their theoretical R-D behaviors. Figure 3.5 shows that the R-D behaviors of both the tree coding schemes are consistent with the theory. Finally, in Figure 3.6, we show the segmentation and model allocation choices made by the prune and prune-join binary tree schemes for the line of the lena image shown in Figure 3.1(b).

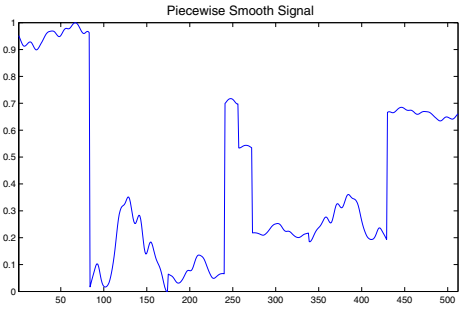
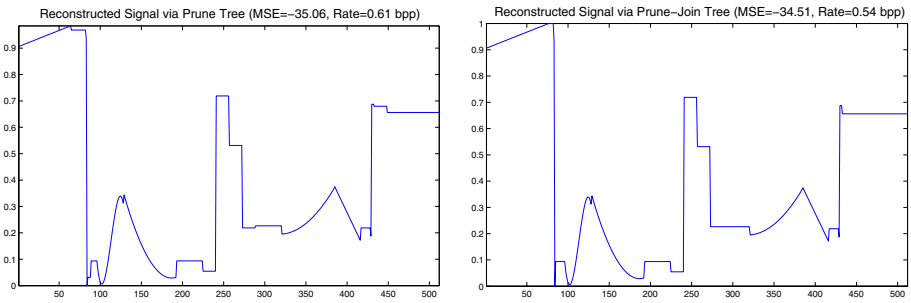


Figure 3.2: Original piecewise smooth signal.



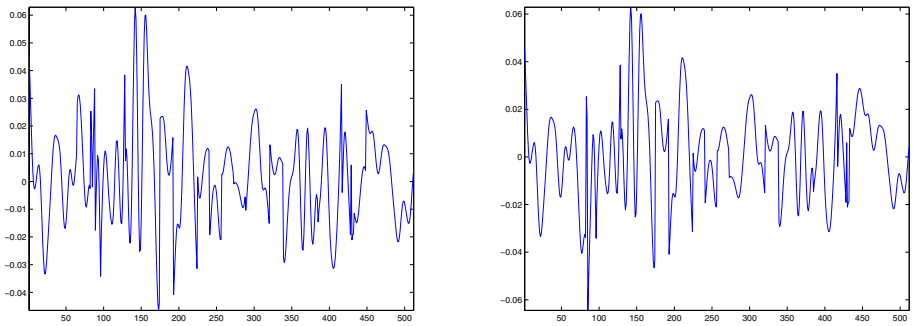
(a) Prune tree (MSE=-35.06 dB, Rate=0.61 bps).

(b) Prune-join tree (MSE=-34.51 dB, Rate=0.54 bps).

Figure 3.3: Approximations provided by the prune and prune-join binary tree coding algorithms.

3.5 Conclusions

In this chapter, we have shown that both the prune and prune-join schemes achieve the near-optimal asymptotic R-D behavior $\left(D(R) \sim d_3 \left(\frac{\log R}{R}\right)^{2\alpha}\right)$ for piecewise smooth signals. The suboptimality factor $(\log R)^{2\alpha}$ is clearly a penalty for using a tree structure. Experimental results shown in Figure 3.5 also confirm the derived R-D behaviors of the tree schemes. Thus, both the theoretical and numerical results show that the tree based schemes can efficiently code piecewise smooth signals, which in turn can efficiently model the real life signals. In the next chapter, we will extend the proposed binary tree schemes to the 2-D scenario using the quadtree segmentation.



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

Figure 3.4: Residual signals provided by the prune and prune-join binary tree coding algorithms.

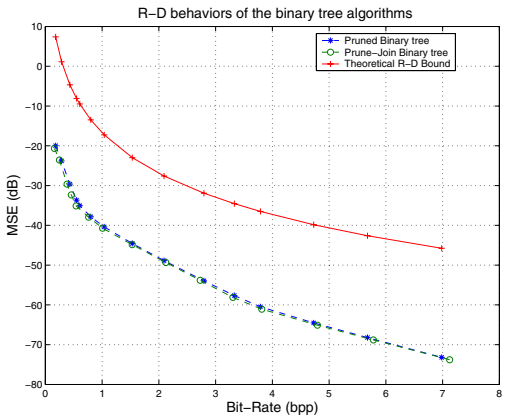
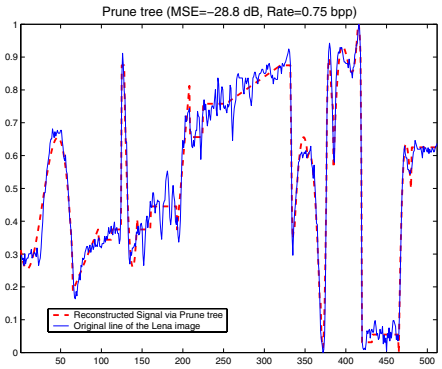
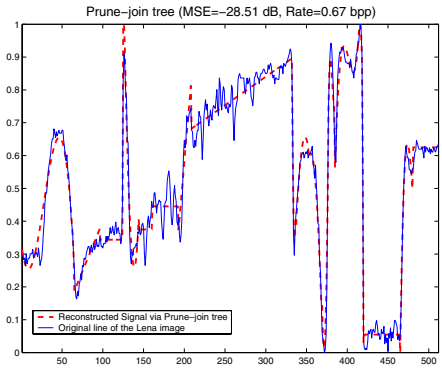


Figure 3.5: R-D performance of the prune and prune-join binary tree algorithms for piecewise smooth signals.



(a) Prune tree (MSE=-28.8 dB, Rate=0.75 bps).



(b) Prune-join tree (MSE=-28.51 dB, Rate=0.67 bps).

Figure 3.6: Piecewise polynomial approximations provided by the prune and prune-join binary tree coding algorithms for a line of the lena image.

Chapter 4

Quadtree Segmentation Algorithms and Piecewise Polynomial Images

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.[1] 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 [16].

This chapter is organized as follows: Section 4.1 outlines the prune and prune-join quadtree schemes. In Section 4.2, for the sake of simplicity, we consider 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 4.1(a)). In Section 4.2.1, we present the oracle R-D behavior. In Sections 4.2.2 and 4.2.3, 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 (The-

[1] As quadtree segmentation can model only horizontal and vertical edges at dyadic locations.

orem 4.2, Section 4.2.3) with computational ease. Now, in Section 4.3, we consider more complex piecewise polynomial image model, where the edge is also a piecewise polynomial curve. We derive the asymptotic R-D behaviors of the proposed quadtree schemes. Again, we prove that the prune-join quadtree scheme achieves the oracle like asymptotic R-D performance. In Section 4.5, we present simulation results, which show the superiority of the proposed quadtree based image coding scheme over a wavelet based coder (JPEG2000) at low bit rates. Finally, we draw conclusions in Section 4.6.

4.1 Quadtree Algorithms

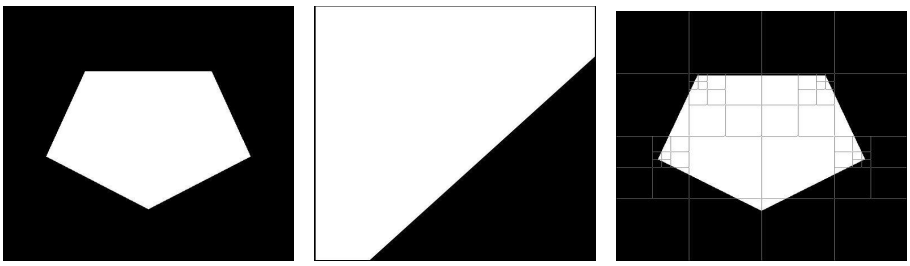
Similar to the 1-D case, we first consider the prune quadtree algorithm. The overall structure of this scheme is similar to the prune binary tree algorithm as described in Algorithm 2.1. Basically, this algorithm employs a quadtree segmentation, which approximates each node by an edge tile consisting of two 2-D polynomials separated by a polynomial boundary.[2] 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 (see Figure 4.1(a)). 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 4.1(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 4.1(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

[2] In general, an edge tile contains a *polynomial* boundary (Section 4.3). However, we use edge tiles with *linear* boundary for the polygonal model (Section 4.2), for the complexity analysis (Section 4.4), and for the numerical experimentation (Section 4.5).



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

Figure 4.1: Examples of a black and white (B/W) polygonal image, an edge tile with a linear boundary, and the quadtree segmentation.

to that of the 1-D case, except that the search algorithm for 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

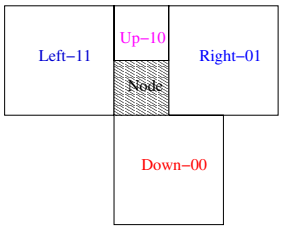


Figure 4.2: 4-connected neighboring nodes. Every neighbor is assigned a two bit index.

that their origins (bottom-left points) are (x_1, y_1) and (x_2, y_2), respectively. Figure 4.2 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.

This search scheme also highlights that the 2-D scenario is much more complex than the 1-D case.

4.2 R-D Analysis for the Polygonal Image Model

4.2.1 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

$$D(R) \leq V 2^{-R/2V}. \tag{4.1}$$

In the next two Sections 4.2.2 and 4.2.3, we present the R-D performance of the two quadtree algorithms for the polygonal model. Similar to 1-D, we provide

only upper-bounds for operational R-D functions of the quadtree algorithms. The reason is that this analysis provides us the insight into the R-D performance of the algorithms in the high rate regime. Another reason is that it is difficult to compute the exact R-D function in a general setting such as ours.

4.2.2 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 4.1, which states the suboptimal R-D behavior of the prune quadtree scheme.

Lemma 4.1 *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 with linear boundary. Since the polygon has V vertices, there are at most V splitting nodes at each tree 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

$$N_0 \quad \leq \quad (J-1)3V + 4V = (3J+1)\,V. \tag{4.2}$$

◇

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

$$M_0 \quad \leq \quad 4JV + 1. \tag{4.3}$$

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.

Theorem 4.1 *For the polygonal model, the prune quadtree coding algorithm, which employs the parent-children pruning, results in the following asymptotic R-D behavior*

$$D_P\,(R) \leq c_6 \sqrt{R} 2^{-c_7\sqrt{R}}, \tag{4.4}$$

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

Proof: Since the polygonal image has V vertices, at most V leaves will have a vertex. The remaining $3JV$ leaves (Lemma 4.1) can be simply represented by edge tiles. If J is large enough, then in the worst case each vertex will lie in a different dyadic square leaf at 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 $(0, 1)$. The remaining leaves without vertices can be well approximated by edge tiles. For coding an edge block with a linear-edge, we need to code the locations of two vertices of the linear-edge on the boundary of the block. 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 particular, an edge leaf l at tree level j is of side-length $T_l = 2^{-j}$. In this case, if we allocate $\frac{R_l}{2}$ bits to each line-vertex of the linear edge of the edge leaf, then the maximum distance between the true line vertices and their quantized version is bounded by $2^{-j}2^{-(\frac{R_l}{2}-1)}$. Thus, the distortion of an edge leaf will be bounded by $2^{-2j}2^{-(\frac{R_l}{2}-1)}$.[3] Since R-D optimal solution of exponentially decaying R-D functions results in equal distortion for each leaf [12], the coding algorithm will allocate same rate R_j to all the leaves without vertices at the same tree level j.

Similar to 1-D, R-D optimization results in a tree-depth J and a bit allocation strategy such that the edge 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 $2(2(J-j)+1)$ bits to every edge block at tree level j to ensure that the distortion for each leaf is bounded by 2^{-2J}. Since all the tree levels, except $j = 0$, can contribute $3V$ leaves with no vertex, the total rate required for coding the leaves is

$$R_{\text{Leaves}} = 3V\sum_{j=1}^{J} 2(2(J-j)+1) = 6VJ^2. \tag{4.5}$$

The tree split-merge decision variable will consume bits (R_{Tree}) equal to the total number of nodes in the pruned tree. Thus, (4.3) provides $R_{\text{Tree}} \leq 4JV+1$. Hence, the total bitrate can be written as follows

$$\begin{aligned} R &= R_{\text{Tree}} + R_{\text{Leaves}} \\ R &\leq 6VJ^2 + 4JV + 1 \leq 6V(J+1)^2. \end{aligned} \tag{4.6}$$

Since all the leaves have a distortion of the same order $O\left(2^{-2J}\right)$, the net distortion can be bounded as follows

$$\begin{aligned} D &= 3JV\left(2^{-2J}\right) + V\left(\frac{1}{4}2^{-2J}\right) \leq 3V(J+1)2^{-2J} \\ \Rightarrow D &\leq 2\sqrt{6VR}2^{-\sqrt{\frac{2}{3V}R}}; \text{ combining (4.6) and (4.7).} \end{aligned} \tag{4.7}$$

Thus, the prune quadtree algorithm performs suboptimally. □

[3] This distortion bound will be achieved if the linear edge is the diagonal of the dyadic square.

4.2.3 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 2.4, one can prove the following lemma:

Lemma 4.2 *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 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 leaves with vertices will not be coded, the number of the encoded leaves becomes V.

◇

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

Theorem 4.2 *For the polygonal model, the prune-join quadtree algorithm, which jointly encodes similar neighbors, achieves the oracle like exponentially decaying asymptotic R-D behavior*

$$D_{PJ}(R) \leq c_8 2^{-c_9 R}, \tag{4.8}$$

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 a

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}.[4] 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, (4.3) 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 (4.2)). Hence, the total bitrate is as follows

$$\begin{aligned} R &= R_{\text{Tree}} + R_{\text{LeafJointCoding}} + R_{\text{Leaves}} \\ R &\leq 17\left(J + \frac{1}{2}\right)V; \text{ as } V > 0. \end{aligned} \tag{4.9}$$

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

$$D_{PJ} = V\left(2^{-2J}\right) + V\left(\frac{1}{4}2^{-2J}\right) \tag{4.10}$$

Combining (4.9) and (4.10) provides

$$D_{PJ} \leq \frac{5}{2}V2^{-\frac{2}{17V}R}. \tag{4.11}$$

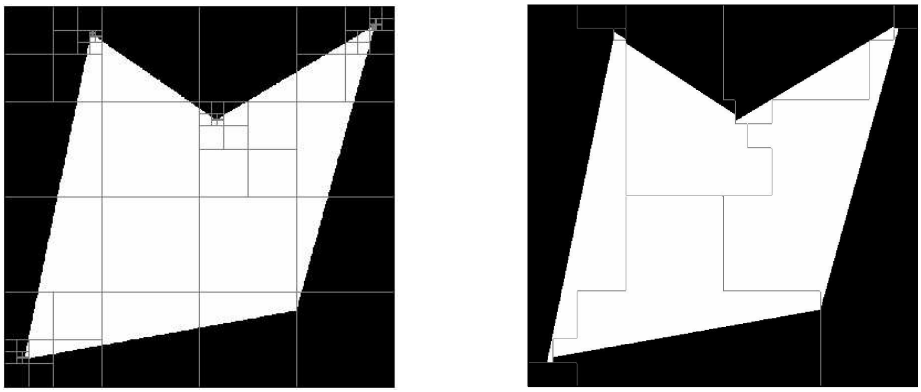
Therefore, the prune-join tree algorithm achieves an exponentially decaying R-D behavior. □

An example of the two schemes is shown in Figure 4.3. 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 complexity.

4.3 R-D Analysis for the Piecewise Polynomial Image Model

In this section, we analyze the R-D performance for more general piecewise polynomial image model, where the edge is also a piecewise polynomial curve. First, we present the oracle R-D performance. We then derive the upper-bounds for the R-D performance of the quadtree algorithms.

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



(a) Prune tree segmentation.

(b) Prune-join tree segmentation.

Figure 4.3: Examples of the quadtree representation for the polygonal model.

4.3.1 Oracle R-D Performance

A continuous-space 2-D piecewise polynomial signal $f(x, y)$, $(x, y) \in [0, T]^2$ with M 2-D polynomial pieces of degree $\leq P$, can be defined in the following way

$$f(x, y) = \sum_{i=1}^{M} p_i(x, y)\mathbf{1}_{\mathcal{R}_i}(x, y), \tag{4.12}$$

where $\bigcup_{i=1}^{M} \mathcal{R}_i = [0, T]^2$ and $p_i(x, y)$ is the 2-D polynomial associated with the region $\mathcal{R}_i$. Moreover, the region boundaries are composed of 1-D polynomials of degree $\leq Q$. Suppose that the total number of 1-D polynomial boundaries is S and the polynomial boundaries intersect each other at V points, which we call vertices.[5] Assume that the function $f(x, y)$ and the 1-D polynomial boundaries are bounded in magnitude by some constants A and $A_S > 1$, respectively. To simplify the analysis, we further assume that any 1-D polynomial boundary starts from a vertex or a side of the bounding box of the whole image and ends with its first encounter with a vertex or a side of the bounding box. Thus, a 1-D polynomial boundary can have no more than 2 vertices and can separate only two 2-D polynomial regions. Figure 4.4 shows a 2-D piecewise polynomial image which contains 17 polynomial boundaries because the thick violet line is treated as 3 separate boundaries in the proposed scenario.

In the following oracle based R-D analysis for 2-D piecewise polynomials, we first determine a general R-D bound for the single 2-D polynomial piece. We then compute an R-D bound for encoding a polynomial boundary. Finally, we

[5]Note that the total number V of vertices is equal to the total number S of 1-D boundaries for the polygonal model.

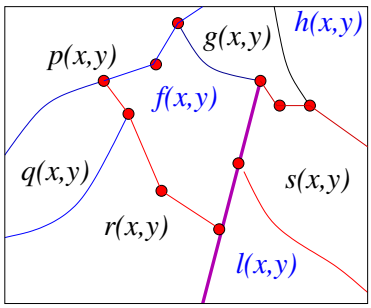


Figure 4.4: An example of piecewise polynomial image with piecewise polynomial boundaries. Red dots indicate the vertices of piecewise polynomial singularities. The shown image contains 10 vertices.

present the jointly optimal bit allocation along with the simplified global R-D upper-bound for the whole signal.

R-D analysis of one polynomial piece

Consider a P^{th} order polynomial $q(x, y)$ defined over the region $\mathcal{R} = [a, b] \times [c, d]$. Assume that the polynomial is bounded in magnitude by some constant A. In the local orthogonal Legendre basis expansion, the polynomial $q(x, y)$ can be expressed as follows

$$\begin{aligned} q(x,y) &= \sum_{i=0}^{P} \sum_{j=0}^{P-i} a_{ij} x^i y^j \\ &= \sum_{i=0}^{P} \sum_{j=0}^{P-i} \frac{(2i+1)(2j+1)}{(b-a)(d-c)} l_{ij} L_{\mathcal{R}}(i,j;x,y), \end{aligned} \tag{4.13}$$

where $L_{\mathcal{R}}(i, j; x, y)$ is the $(i + j)^{th}$ degree Legendre polynomial over $\mathcal{R}$ constructed by the product of the 1-D Legendre polynomials[6]

$$L_{\mathcal{R}}(i,j;x,y) = L_{[a,b]}(i;x) L_{[c,d]}(j;y). \tag{4.14}$$

Due to the properties of the Legendre basis expansion, the Legendre coefficients l_{ij} can be bounded as follows

$$|l_{ij}| \leq A(b-a)(d-c); \quad 0 \leq i, j, i+j \leq P. \tag{4.15}$$

The squared error distortion after quantizing the coefficients can be written

[6] Readers are referred to [41, Chapter 3] for the description of local orthogonal Legendre basis expansion in 1-D.

as follows

$$\begin{aligned} D_q &= \sum_{i=0}^{P}\sum_{j=0}^{P-i}\left(\frac{(2i+1)(2j+1)}{(b-a)(d-c)}\right)^2\left(l_{ij}-\widehat{l}_{ij}\right)^2\int_{\mathcal{R}}L^2_{\mathcal{R}}(i,j;x,y) \\ &= \sum_{i=0}^{P}\sum_{j=0}^{P-i}\frac{(2i+1)(2j+1)}{(b-a)(d-c)}\left(l_{ij}-\widehat{l}_{ij}\right)^2. \end{aligned} \tag{4.16}$$

where $\widehat{l}_{ij}$ are the quantized coefficients. Suppose that for each coefficient a different r_{ij} bit uniform quantizer over the range $A(b-a)(d-c)$ with a step size of $2A(b-a)(d-c)2^{-r_{ij}}$ is used. Thus, the distortion for a polynomial piece can be bounded as follows

$$\begin{aligned} D_q &\leq A^2(b-a)(d-c)\sum_{i=0}^{P}\sum_{j=0}^{P-i}(2i+1)(2j+1)2^{-2r_{ij}} \\ D_q &\leq D_q^B = A_0\sum_{i=0}^{P}\sum_{j=0}^{P-i}(2i+1)(2j+1)2^{-2r_{ij}};\ \ A_0 = A^2(b-a)(d-c). \end{aligned}$$

For a global bit budget of R_q bits, with R_q sufficiently large, the optimal bit allocation for minimizing the distortion bound D_q^B can be found by solving the reverse water-filling problem

$$\begin{cases} \dfrac{\partial D_q^B}{\partial r_{ij}} = \text{constant} \\ \\ \sum r_{ij} = R_q \end{cases} \tag{4.17}$$

which results into

$$r_{ij} = \frac{2R_q}{(P+1)(P+2)} + \frac{1}{2}\log_2\left(\frac{(2i+1)(2j+1)}{\bar{C}}\right), \tag{4.18}$$

with

$$\bar{C} = \left[\prod_{0\leq i,j,i+j\leq P}(2i+1)(2j+1)\right]^{\frac{2}{(P+1)(P+2)}} \tag{4.19}$$

$$\bar{C} \leq \frac{(P+1)(P+2)}{2};\ \text{as geometric mean} \leq \text{arithmetic mean}. \tag{4.20}$$

Therefore, the distortion for a single polynomial piece can be bounded as follows

$$D_q(R_q) \leq A_0\left[\frac{(P+1)(P+2)}{2}\right]^2 2^{-\frac{4}{(P+1)(P+2)}R_q}. \tag{4.21}$$

R-D analysis of one polynomial boundary, which separates 2-D polynomial pieces

Now, we will analyze the rate-distortion function for encoding one polynomial boundary $s(x)$ of degree Q, whose coefficients are precisely provided by an oracle.

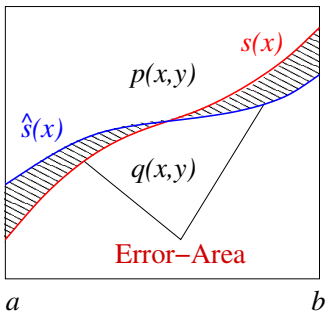


Figure 4.5: Region of error due to encoding of polynomial boundary with certain bit rate.

If we quantize the polynomial boundary only, then the distortion for the 2-D signal is incurred only in the shaded region (error-area) shown in Figure 4.5.

As 2-D polynomial mismatch occurs only in the shaded region, the coding distortion for the polynomial boundary is as follows

$$\begin{aligned} D_s &= \int_{\text{Error-Area}} (p(x,y) - q(x,y))^2 \\ &\leq (2A)^2 \int_{[a,b]} |s(x) - \widehat{s}(x)|\,; \text{ as } |p(x,y)|, |q(x,y)| \leq A. \\ &\leq (2A)^2 \sqrt{\int_{[a,b]} |s(x) - \widehat{s}(x)|^2 \, (b-a)}; \text{ by Schwartz's inequality.} \end{aligned} \quad (4.22)$$

If we encode the polynomial boundary using R_s bits, then we know from [41] that

$$\int_{[a,b]} |s(x) - \widehat{s}(x)|^2 \leq A_S^2 (b-a) \, (Q+1)^2 \, 2^{-\frac{2}{Q+1} R_s}. \quad (4.23)$$

Combining (4.22) and (4.23) provides

$$D_s \, (R_s) \leq 4A^2 A_S (b-a)(Q+1) 2^{-\frac{1}{Q+1} R_s}. \quad (4.24)$$

We can now compute the global R-D bound for piecewise polynomial signals with piecewise polynomial boundaries.

Global R-D bound

The signal is uniquely determined by M 2-D polynomials and by S 1-D polynomial 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 2-D polynomial pieces and the 1-D polynomial singularities are provided with arbitrary accuracy by an oracle. We can simply represent a 2-D piecewise polynomial by encoding the region-boundaries and associated 2-D polynomials. Since the region boundaries are piecewise polynomials and any polynomial

boundary will be part of two region boundaries, an efficient way to encode a region boundary is just to specify which 1-D polynomial boundaries are forming the region boundary in an ordered manner such as clockwise. That means, we encode all the 1-D polynomial singularities using the standard scalar quantization procedure and assign unique indices to 1-D polynomial singularities.[7] Now, the region boundary will be encoded by these indices of the 1-D polynomial singularities, which form the region boundary. Let us further assume that the region of support $\mathcal{R}_i$ for the polynomial piece $p_i(x,y)$ of degree P_i is enclosed in the rectangular region $[a_i,b_i]\times[c_i,d_i]$. Thus, if we allocate R_{p_i} bits to the 2-D polynomial, then (4.21) provides the following distortion bound due to the quantization of 2-D polynomial coefficients

$$D_{p_i}\left(R_{p_i}\right) \leq A^2\left(b_i-a_i\right)\left(d_i-c_i\right)\left[\frac{(P_i+1)(P_i+2)}{2}\right]^2 2^{-\frac{4}{(P_i+1)(P_i+2)}R_{p_i}}.$$

Similarly, suppose that j^{th} polynomial boundary $s_j(x)$ is defined over the interval $[a_j,b_j]$ and has the degree Q_j. So if we allocate R_{s_j} bits to the 1-D polynomial boundary, then (4.24) results into the following distortion bound due to the quantization of 1-D polynomial boundary coefficients

$$D_{s_j}\left(R_{s_j}\right) \quad \leq \quad 4A^2A_S(b_j-a_j)(Q_j+1)2^{-\frac{1}{Q_j+1}R_{s_j}}.$$

The global distortion bound for the whole signal will be the sum of the quantization distortion of M 2-D polynomial pieces and the quantization distortion of S polynomial boundaries. Thus, the global distortion bound can be written as follows

$$D \quad \leq \quad \sum_{i=1}^{M} D_{p_i}\left(R_{p_i}\right)+\sum_{j=1}^{S} D_{s_j}\left(R_{s_j}\right). \tag{4.25}$$

However, this R-D function is very complicated and depends on all the polynomial parameters across the whole function. So we make some assumptions to derive the simplified R-D bound and associated bit allocation strategy:

- All 2-D polynomial pieces and 1-D polynomial boundaries are assumed of maximum degree P and Q, respectively. This simply means that for a polynomial of lower degree bits are also allocated to zero coefficients.
- The region of support for each 2-D polynomial piece and 1-D polynomial boundary is approximated by $[0,T]^2$ and $[0,T]$ respectively.
- The bounding box $[0,T]^2$ of the whole function is known. Thus, we need not code the bounding box boundary.

[7] Since the image has S polynomial boundaries and 4 boundaries of the bounding box of the image, we need $\lceil \log_2(S+4) \rceil$ bits to code an index associated with a polynomial boundary. Moreover, vertex information can be simply obtained by solving the intersection equations of polynomial boundaries. So each vertex information requires $\lceil \log_2 Q \rceil$ bits as side information.

These assumptions ensure that the reverse water filling R-D optimization will allocate the same number of bits R_p to each 2-D polynomial and R_s bits to each polynomial boundary. Therefore, the simplified R-D bound becomes

$$D(R) \leq MA^2T^2 \left[\frac{(P+1)(P+2)}{2}\right]^2 2^{-\frac{4}{(P+1)(P+2)}R_p} + S4A^2A_ST(Q+1)2^{-\frac{1}{Q+1}R_s} \tag{4.26}$$

where the total bit rate is $R = MR_p + SR_s$.[8] Now, by the usual reverse water filling technique, we obtain the following optimal bit allocation

$$R_p = \frac{(P+1)(P+2)}{M(P+1)(P+2)+4S(Q+1)}R + S \cdot C \tag{4.27}$$

$$R_s = \frac{4(Q+1)}{M(P+1)(P+2)+4S(Q+1)}R - M \cdot C, \tag{4.28}$$

where $C = \frac{(P+1)(P+2)(Q+1)}{M(P+1)(P+2)+4S(Q+1)} \log_2 \left(\frac{M(P+1)(P+2)}{4SA_S}\right)$.

Combining (4.26), (4.27) and (4.28), the simplified global R-D bound becomes

$$D(R) \leq 2MA^2T^2(P+1)^2(P+2)^2SA_S 2^{-\frac{4}{M(P+1)(P+2)+4S(Q+1)}R}. \tag{4.29}$$

This concludes that the oracle based method attains an exponentially decaying R-D behavior for 2-D piecewise polynomial functions with piecewise polynomial boundaries. This result can be more formally stated as the following theorem:

Theorem 4.3 *An oracle based method, which simply scalar quantizes the region boundaries and associated 2-D polynomials, achieves the following exponentially decaying R-D behavior*

$$D(R) \leq c_{10}2^{-c_{11}R}, \tag{4.30}$$

where $c_{10} = 2MA^2T^2(P+1)^2(P+2)^2SA_S$ *and* $c_{11} = \frac{4}{M(P+1)(P+2)+4S(Q+1)}$*, for 2-D piecewise polynomial images as defined in (4.12).*

4.3.2 R-D Analysis of the Prune Quadtree Algorithm

In this section, we present the R-D performance of the prune quadtree algorithm for 2-D piecewise polynomial signals as defined in (4.12). The presented R-D analysis closely follows the logic outlined for the polygonal image model. We first show that the prune quadtree scheme encodes a number of leaves, which grows linearly with respect to the tree depth J. We then present Theorem 4.4, which states the suboptimal R-D behavior of the prune quadtree scheme.

Lemma 4.3 *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 (intersection points of boundaries) in the 2-D piecewise polynomial signal, respectively.*

[8] In this simplified calculation, we have ignored the side information cost, like the coding cost of indices of polynomial boundaries, as this cost is negligible at high rates.

Proof: Since the prune quadtree algorithm approximates each node by an edge tile consisting of two 2-D polynomials separated by a polynomial boundary, it can efficiently model only those nodes which do not contain vertices of piecewise polynomial boundary. Therefore, at high rates, the prune quadtree segmentation scheme recursively divides only those dyadic blocks which contain a vertex of the edge. Other dyadic blocks contain either no edge or a polynomial edge, so they can be efficiently represented by the edge tiles.

Since the 2-D piecewise polynomial signal has V vertices, there are at most V splitting nodes at each tree level. Thus, they will generate no more than $3V$ leaves with an 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

$$N_0 \quad \leq \quad (J-1)3V + 4V = (3J+1)\,V. \tag{4.31}$$

$\diamond$

Since every tree level can have at most $4V$ nodes, the total number M_0 of nodes in the pruned quadtree can be given by

$$M_0 \quad \leq \quad 4JV + 1. \tag{4.32}$$

Even if the 2-D piecewise polynomial image has a finite number of degrees of freedom, the prune quadtree scheme encodes a number of parameters which grows linearly with J. Hence, it is bound to exhibit a suboptimal R-D behavior. We present this result as the following theorem.

Theorem 4.4 *The prune quadtree coding algorithm, which employs the R-D optimal parent-children pruning, results in the following asymptotic R-D behavior*

$$D_P\,(R) \leq c_{12}\sqrt{R}2^{-c_{13}\sqrt{R}}, \tag{4.33}$$

where $c_{12} = \frac{48\sqrt{2}VA^2T^2(P+1)^2(P+2)^2A_S}{\sqrt{3V((P+1)(P+2)+2(Q+1))}}$ *and* $c_{13} = \sqrt{\frac{8}{3V((P+1)(P+2)+2(Q+1))}}$*, for 2-D piecewise polynomial signals as defined in (4.12).*

Proof: Suppose that the quadtree is decomposed up to the depth J. Clearly, the prune tree scheme splits only those tree nodes which contain vertices of the piecewise polynomial boundary. Since the 2-D piecewise polynomial image has only V vertices, at most V leaves will have a vertex. The remaining $3JV$ leaves (Lemma 4.3) can be simply represented by edge tiles. In the worst case, at high rates, each vertex will lie in a different dyadic leaf at the depth J. As their sizes will be $T^2 2^{-2J}$, the distortion of each of the vertex containing leaves is bounded by $A^2T^2 2^{-2J}$. Leaves without vertices can be well approximated by an edge tile. In particular, a leaf l at tree level j is of side-size $T_l = T2^{-j}$ and its R-D function can be bounded by $4A^2T^2 2^{-2j}(P+1)^2(P+2)^2A_S 2^{-\frac{4}{2(P+1)(P+2)+4(Q+1)}R_l}$ (By putting $M = 2, S = 1, T = T_l, R = R_l$ in (4.29)). Since R-D optimal solution of exponentially decaying R-D functions results in equal distortion for

each leaf [12], the coding algorithm will allocate the same rate R_j to all the leaves without vertices at the same tree level j.

Similar to 1-D, R-D optimization selects a tree-depth J and a bit allocation strategy such that the edge leaves and the vertex containing leaves have a distortion of the same order $O\left(2^{-2J}\right)$. Therefore, the coding algorithm will allocate $[(P+1)(P+2)+2(Q+1)](J-j)$ bits to each of the leaves with no vertex at level j and no bits to leaves with vertices.[9] This ensures that all the leaves have a distortion of the same order $O\left(2^{-2J}\right)$. Hence, the net distortion can be bounded as follows

$$\begin{aligned} D_P &= V\left(A^2T^2 2^{-2J}\right) + 3JV\left(4A^2T^2(P+1)^2(P+2)^2A_S 2^{-2J}\right) \\ \Rightarrow D_P &\leq 12VA^2T^2(P+1)^2(P+2)^2A_S(J+1)2^{-2J}. \end{aligned} \tag{4.34}$$

Since all the tree levels, except $j=0$, can contribute $3V$ leaves with no vertex, the total rate required for coding the leaves is

$$\begin{aligned} R_{\text{Leaves}} &= 3V\sum_{j=1}^{J}[(P+1)(P+2)+2(Q+1)](J-j) \\ &= 3V[(P+1)(P+2)+2(Q+1)]\frac{J(J-1)}{2}. \end{aligned} \tag{4.35}$$

The tree split-merge decision variable will consume bits (R_{Tree}) equal to the total number of nodes in the pruned tree. Thus, (4.3) gives $R_{\text{Tree}} \leq 4JV+1$. The total bit rate can be seen as the sum of the costs of coding the tree itself and the quantized model parameters of the leaves. Hence, the total bit rate can be written as follows

$$\begin{aligned} R &= R_{\text{Tree}} + R_{\text{Leaves}} \\ &\leq 4JV + 1 + 3V[(P+1)(P+2)+2(Q+1)]\frac{J(J-1)}{2} \\ \Rightarrow R &\leq \frac{3V[(P+1)(P+2)+2(Q+1)]}{2}(J+1)^2;\ \text{as } V>0. \end{aligned} \tag{4.36}$$

Combining (4.34) and (4.36) by eliminating J and noting that the right hand side of (4.34) is a decreasing function of J, whereas the right hand side of (4.36) is an increasing function of J, we obtain the following R-D bound

$$D_P \leq \frac{48\sqrt{2}VA^2T^2(P+1)^2(P+2)^2A_S}{\sqrt{3V[(P+1)(P+2)+2(Q+1)]}}\sqrt{R}2^{-\sqrt{\frac{8}{3V[(P+1)(P+2)+2(Q+1)]}R}}.$$

Hence, the prune quadtree scheme performs suboptimally. □

4.3.3 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 for 2-D piecewise polynomial signals. First of all, we prove that the prune-join quadtree scheme encodes a fixed

[9] As any vertex containing leaf has the distortion bounded by $A^2T^2 2^{-2J}$ which will not reduce with the rate.

number of leaves, which is equal to the number of 1-D polynomial singularities S in the 2-D piecewise polynomial image.

Lemma 4.4 *The prune-join quadtree algorithm, which jointly encodes similar neighbors, reduces the effective number of leaves to be encoded to S, where S represent the number of polynomial singularities in the 2-D piecewise polynomial image.*

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 can be well represented by an edge tile. It is also clear that there will be at most V vertices containing leaves 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 has S polynomial boundaries and each boundary can separate only two 2-D polynomials by construction (see Section 4.3.1), the maximum number of distinct edge blocks (leaves) will be S. The neighbor joint coding ensures that all the similar leaves characterized by same edge tile will be joined to form one joint block. Since the image has S different edge tiles, the neighbor joint coding will result into S joint blocks. Therefore, the prune-join tree algorithm provides S joint leaves and V leaves with a vertex. Since the leaves containing a vertex will not be coded, the number of encoded leaves becomes S. Thus, the number of leaves to be coded remains constant with respect to the tree depth J. ◇

We can now state the following theorem:

Theorem 4.5 *The prune-join quadtree algorithm, which jointly encodes similar neighbors, achieves the oracle like exponentially decaying asymptotic R-D behavior*

$$D_{PJ}(R) \leq c_{14} 2^{-c_{15} R}, \tag{4.37}$$

where $c_{14} = 4A^2T^2\left[4S(P+1)^2(P+2)^2A_S + V\right]$ *and* $c_{15} = \frac{2}{(S+V)[(P+1)(P+2)+2(Q+6)]}$, *for 2-D piecewise polynomial signals as defined in (4.12).*

Proof: The prune-join quadtree algorithm provides S joint blocks to be encoded and at most V leaves with a vertex. In the worst case, each vertex will lie in a different dyadic leaf at the depth J. As their sizes will be $T^2 2^{-2J}$, the distortion of each of the vertex containing leaves is bounded by $A^2T^2 2^{-2J}$ and it does not decrease with the rate (recall that the algorithm tries to approximate each block with an edge tile). The size of each joint block can be bounded by the whole image size T^2. Thus, the distortion of each joint block is bounded by $4A^2T^2(P+1)^2(P+2)^2A_S 2^{-\frac{4}{2(P+1)(P+2)+4(Q+1)}R_l}$, where R_l is the rate allocated to that block. Again, R-D optimization forces all the blocks to have the distortion of the same order. Therefore, the algorithm will allocate $[(P+1)(P+2)+2(Q+1)]\,J$ bits to each joint block and no bits to the leaves with singularities. This ensures that all the leaves have the distortion of the same order $O\left(2^{-2J}\right)$.

As there are only S joint blocks, the bitrate required for coding the leaves is $R_{\text{Leaves}} = S\left[(P+1)(P+2)+2(Q+1)\right]J$. In the prune-join coding scheme, the side information is composed of two parts: 1. Bits required to code the pruned tree: $R_{\text{Tree}} \leq 4JV+1$. 2. Bits required to code the leaf joint coding tree ($R_{\text{LeafJointCoding}}$). 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 $R_{\text{LeafJointCoding}} \leq 3 \times (3J+1)V$ (from (4.2)). Hence, the total bitrate can be written as follows

$$\begin{aligned} R &= R_{\text{Tree}} + R_{\text{LeafJointCoding}} + R_{\text{Leaves}} \\ R &\leq (S+V)\left[(P+1)(P+2)+2(Q+6)\right](J+1);\ \text{as } S,V>0 \end{aligned} \tag{4.38}$$

The overall distortion is the sum of the distortions of S joint blocks and V leaves with a vertex and it can be written as follows

$$\begin{aligned} D_{PJ} &= S\left(4A^2T^2(P+1)^2(P+2)^2A_S 2^{-2J}\right) + V\left(A^2T^2 2^{-2J}\right) \\ &= A^2T^2\left(4S(P+1)^2(P+2)^2A_S + V\right)2^{-2J} \end{aligned} \tag{4.39}$$

Combining (4.38) and (4.39), we obtain the following R-D bound

$$D_{PJ} \leq 4A^2T^2\left[4S(P+1)^2(P+2)^2A_S+V\right]2^{-\frac{2}{(S+V)[(P+1)(P+2)+2(Q+6)]}R}.$$

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

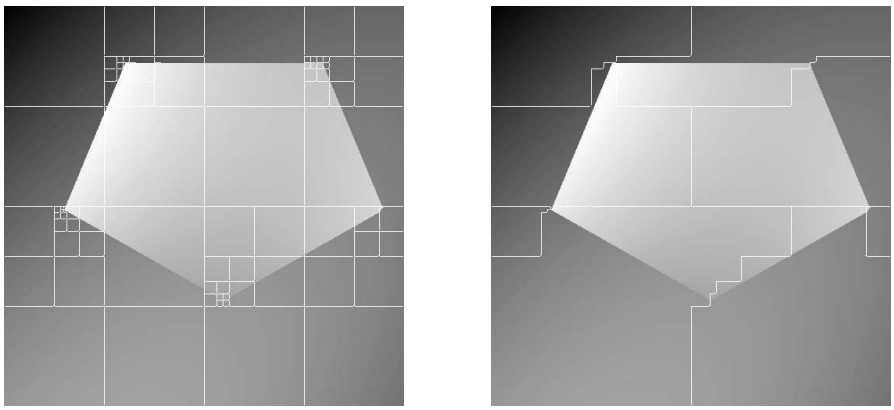
Figure 4.6 shows the segmentation performed by the proposed quadtree schemes. We can clearly see that the prune tree scheme recursively divides the same edge to capture a vertex, while the prune-join scheme successfully joins the similar neighbors as well as captures the vertices.

4.4 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 $O(\log n)$.

1. *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 for a tree node. In 2-D, a P^{th} order polynomial $p(x,y)$ over a region Ω is defined as follows:[10]

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



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

Figure 4.6: Segmentation performed by the quadtree algorithms for a piecewise quadratic image.

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

where $\phi_{ij}(x,y), 0 \le i,j,i+j \le P$, are the 2-D Legendre polynomial basis functions over the region Ω and $l_{ij}, 0 \le i,j,i+j \le P$, 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 $\{x^i y^j, 0 \le i,j,i+j \le P\}$. 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 $\widehat{\Omega} = \{(x_k, y_k), 1 \le k \le L\}$,[11] the minimum squared-error Legendre polynomial approximation $\boldsymbol{p}$ of order P is obtained by solving the least square (LS) problem:

$$\min_{\boldsymbol{p}} \|V_{L,P}\,\boldsymbol{p} - \boldsymbol{z}\|^2, \tag{4.40}$$

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:

$$V_{L,P} = \begin{bmatrix} \phi_{00}(x_1,y_1)\cdots\phi_{0P}(x_1,y_1)\cdots\phi_{m0}(x_1,y_1)\cdots\phi_{m(P-m)}(x_1,y_1)\cdots\phi_{P0}(x_1,y_1) \\ \phi_{00}(x_2,y_2)\cdots\phi_{0P}(x_2,y_2)\cdots\phi_{m0}(x_2,y_2)\cdots\phi_{m(P-m)}(x_2,y_2)\cdots\phi_{P0}(x_2,y_2) \\ \cdots \\ \phi_{00}(x_L,y_L)\cdots\phi_{0P}(x_L,y_L)\cdots\phi_{m0}(x_L,y_L)\cdots\phi_{m(P-m)}(x_L,y_L)\cdots\phi_{P0}(x_L,y_L) \end{bmatrix} \tag{4.41}$$

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

Similar to 1-D, the solution to the LS problem in (4.40) 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.

2. *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 orientations like the wedgelet/beamlet dictionary [17].[12] 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 (4.41). 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.[13] 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 cost for 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 2.6 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 4.1 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.

[12] To achieve the theoretical R-D performance for the piecewise polynomial image 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. 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.

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

Table 4.1: Summary of the properties of the different algorithms.

	R-D behavior			
Signal class	Wavelet coder	DP coder	Prune coder	Prune-join 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 PPS	$d_2\frac{\log R}{R}$ [14]	NA	$c_6\sqrt{R}2^{-c_7\sqrt{R}}$	$c_8 2^{-c_9 R}$
	Computational cost			
1-D PPS	$O(NR_Q \log N)$	$O\left(N^3 R_Q\right)$	$O(NR_Q \log N)$	$O(NR_Q \log N)$
2-D PPS	$O(NR_Q \log N)$	NA	$O(NR_Q \log N)$	$O(NR_Q \log N)$

4.5 Simulation Results and Discussion

Numerical experiments are performed for two image classes: 1) Piecewise polynomial image model with polygonal edge, where the polygon's vertices are generated randomly using uniform distribution on the space $[0,1]^2$ and 2-D polynomial coefficients are also produced randomly using uniform distribution on $[-1,1]$. 2) Real images.

For piecewise polynomial images as well as 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 models. Thus, the algorithm will compute $(P+1)$ smooth and $(P+1)^2$ edge models for each tree node.[14] For the given bit budget, the algorithm selects the model with minimum 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 a uniform scalar quantizer to code the 2-D Legendre polynomial coefficients. But, for real images, we need to use a non-uniform quantizer [24] 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 a uniform quantizer, as this coefficient is the representative of the mean value of the image segment, which is basically uniformly distributed. 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

[14] 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.

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 4.7 indicate that the algorithm opts for low order polynomial models at low rates. Figure 4.7 also shows the complex geometrical tiling obtained by the prune-join tree scheme to capture the geometry of the cameraman image.

For the cameraman image, simulation results show that our scheme prefers piecewise linear models over piecewise quadratic models at rates less than 0.2 bpp. Even at higher rates, we gain only 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, but using higher order polynomial boundary model may improve the R-D performance at high rates.

The experimental results shown in Figure 4.8, for the polygonal model, confirm the derived theoretical R-D behaviors. In Figures 4.9, 4.11 and 4.12, we compare the prune-join coding scheme with JPEG2000. Residual images shown in Figure 4.10 also demonstrate that the prune-join scheme captures the image geometry more efficiently in comparison to JPEG2000. Figure 4.10(a) also shows that the residual image obtained by the tree coding scheme essentially contains only the texture part of the cameraman image. Figure 4.13 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 4.13 (a), (b)), the prune-join scheme gives less artifacts. In the textured region (see Figures 4.13 (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 4.2 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.

4.6 Conclusions

In this chapter, we have extended our binary tree based coding algorithm 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 an exponentially decaying asymptotic R-D behavior for the piecewise polynomial image model. Numerical simulations (Figure 4.8) also confirm that the algorithm achieves optimal performance if the input image belongs to the image model. In addition, simulations show that our quadtree

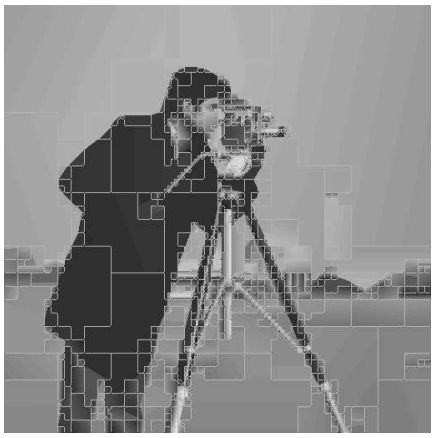


Figure 4.7: Prune-join quadtree tiling for the cameraman image at bitrate=0.071 bpp.

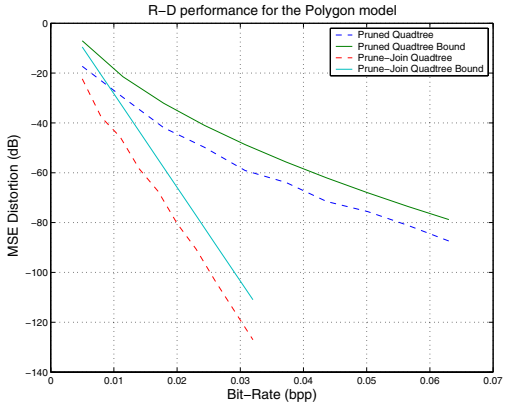


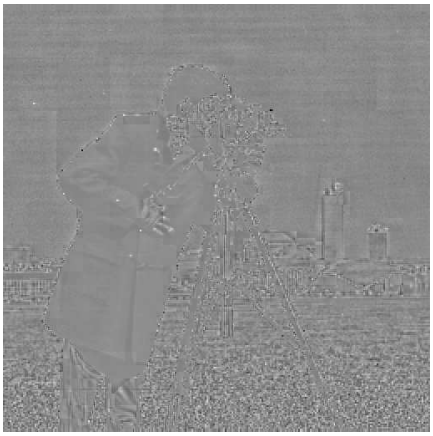
Figure 4.8: Theoretical (solid) and numerical (dotted) R-D curves for the prune and prune-join quadtree algorithms for the polygonal image class.

algorithm consistently outperforms JPEG2000 also in case of compression of real images (Figures 4.9, 4.11, 4.12 and Table 4.2). In brief, the theoretical and experimental results suggest that the quadtree segmentation coupled with piecewise polynomial modeling can efficiently model real life images.

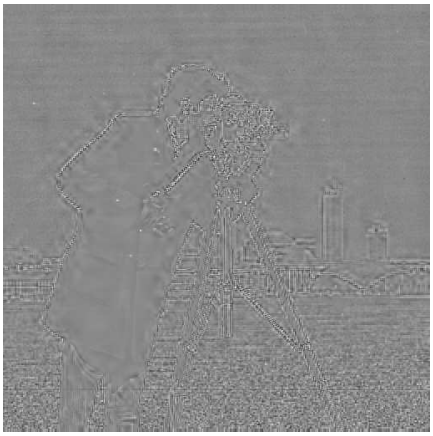
(a) Prune-join quadtree (Rate=0.15 bpp, PSNR=30.68 dB).

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

Figure 4.9: Comparison of the quadtree coder and a wavelet coder (JPEG2000) for the cameraman image.



(a) Prune-join quadtree.



(b) JPEG2000.

Figure 4.10: Residual images of the quadtree coder and JPEG2000 for the cameraman image at 0.15 bpp.

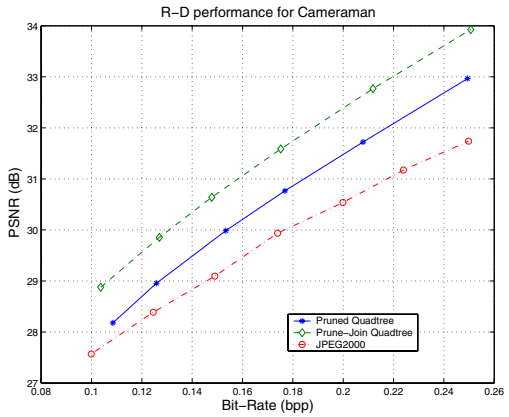


Figure 4.11: R-D performance comparison of the quadtree schemes and JPEG2000 for the cameraman image.

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

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

Figure 4.12: 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.

Figure 4.13: Comparison of artifacts in two regions of the lena image at 0.15 bpp for the prune-join scheme and JPEG2000.

Table 4.2: 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

Chapter 5

Piecewise Smooth Images and Quadtree Segmentation Algorithms

5.1 Introduction

Recent studies have shown that real-life images can be well approximated as a sum of 2-D piecewise smooth functions and textures [24]. In this representation, the piecewise smooth part represents the underlying regular structure of the image, while texture represents the random noise like features of the image. Numerical simulations performed in Section 4.5 of the previous chapter clearly show that if the real image is compressed at low rates, then only the regular structure of the image is captured. This means that a compression scheme must be able to code the regular structure of the image efficiently to achieve higher R-D performance. This prompts us to investigate whether the proposed quadtree based segmentation algorithms can achieve the correct R-D behavior for 2-D piecewise smooth images. In the next section, we consider 2-D smooth functions and show that the quadtree algorithms achieve the oracle like R-D performance (Theorem 5.1). We then analyze the R-D performance for 2-D piecewise smooth functions with smooth boundaries in Section 5.3. In this section, we also prove a strong negative result in the form of Theorem 5.2 which states that, unlike the 1-D case, a 2-D piecewise smooth function cannot be decomposed as a sum of a 2-D piecewise polynomial and a smooth residual function. However, this result poses no obstacle to the proposed tree algorithms in attaining a near-optimal R-D behavior, as shown by Theorem 5.3 and (5.27), for 2-D piecewise smooth functions with smooth boundaries. Section 5.4 shows experimental results, which also confirm the derived R-D behaviors of the quadtree algorithms. Finally, concluding remarks are presented in Section 5.5.

5.2 R-D Analysis for 2-D Smooth Functions

In this analysis, our goal is to compute the R-D behavior of the quadtree algorithms for 2-D smooth functions. We describe the smoothness of a function in terms of the Hölder exponent. Similar to the 1-D case, we can define the smoothness of a function as follows [26, 32, 33]:

Definition 5.1

- A function $f(x,y)$ is pointwise Hölder $\alpha \geq 0$ at (x_0,y_0), if there exist a $K>0$, and a polynomial $p_{(x_0,y_0)}(x,y)$ of degree $P=\lfloor\alpha\rfloor$ such that[1]

$$\forall (x,y)\in\mathbb{R}^2, |f(x,y)-p_{(x_0,y_0)}(x,y)|\leq K\left((\Delta x)^2+(\Delta y)^2\right)^{\frac{\alpha}{2}}, \quad (5.1)$$

  where $\Delta x = x-x_0$ and $\Delta y = y-y_0$.

- A function $f(x,y)$ is uniformly Hölder α over the region Ω if it satisfies (5.1) for all $(x_0,y_0)\in\Omega\subset\mathbb{R}^2$, with a constant K that is independent of (x_0,y_0).

- The Hölder regularity of $f(x,y)$ at (x_0,y_0) or over Ω is the supremum of the α such that $f(x,y)$ is Hölder α.

Furthermore, for a uniformly Hölder α smooth function $f(x,y)$, the polynomial $p_{(x_0,y_0)}(x,y)$ in (5.1) is the P^{th} order Taylor series expansion of $f(x,y)$ around the point (x_0,y_0) and it can be written as follows:

$$\begin{aligned} p_{(x_0,y_0)}(x,y) &= f(x_0,y_0)+\left[f_x(x_0,y_0)\Delta x+f_y(x_0,y_0)\Delta y\right]+ \\ &\quad \frac{1}{2!}\left[f_{xx}(x_0,y_0)(\Delta x)^2+2\Delta x\Delta y f_{xy}(x_0,y_0)+f_{yy}(x_0,y_0)(\Delta y)^2\right]+\cdots \\ &= \sum_{i=0}^{P}\left\{\frac{1}{i!}\left[(x-x_0)\frac{\partial}{\partial x'}+(y-y_0)\frac{\partial}{\partial y'}\right]^i f(x',y')\right\}_{x'=x_0,y'=y_0}. \end{aligned}$$

Thus, the P^{th} order 2-D polynomial function is defined by $\frac{(P+1)(P+2)}{2}$ coefficients associated with the standard basis functions $\left\{x^iy^j; 0\leq i,j,i+j\leq P\right\}$.

Essentially, the regularity of a function guarantees that the function can be well approximated by an appropriate polynomial. Therefore, the proposed quadtree segmentation based coding algorithm utilizes this fact to achieve the desired R-D performance stated in the following theorem.

Theorem 5.1 *For 2-D smooth functions, which are uniformly Hölder α over $[0,T]^2$, the full quadtree decomposition algorithm, which codes every quadtree node at the tree depth by a polynomial approximation, achieves the following asymptotic R-D behavior*

$$D_{Full}(R)\leq d_3\left(\frac{\log R}{R}\right)^{\alpha}, \quad (5.2)$$

where d_3 is a positive constant which depends on the smoothness, magnitude and region of support of the function.

[1] To be more precise, α and P satisfy the following relation: $P<\alpha\leq P+1$.

Proof: Consider a 2-D function $f(x,y)$, which is uniformly Hölder α smooth and defined over the region $[0,T]^2$. The quadtree decomposition leads to the square partitions of side-size $T2^{-J}$ at the depth J. These partitions can be represented by the regions $\mathcal{R}_{ij} = [i,(i+1)]T2^{-J} \times [j,(j+1)]T2^{-J}; 0 \le i,j < 2^J$. The quadtree algorithm approximates the function in each segment $\mathcal{R}_{ij}$ by its best (least square error) polynomial approximation $p_{ij}(x,y)$ of degree $P = \lfloor \alpha \rfloor$. Since $p_{ij}(x,y)$ is the best P^{th} order polynomial approximation of the function $f(x,y)$ on the region $\mathcal{R}_{ij}$, $(f(x,y) - p_{ij}(x,y))$ will be orthogonal to any P^{th} order polynomial $q(x,y)$ on the region $\mathcal{R}_{ij}$. That means,

$$\int_{\mathcal{R}_{ij}} (f(x,y) - p_{ij}(x,y))\, q(x,y) \quad = \quad 0. \tag{5.3}$$

Due to (5.1), the polynomial approximation squared error for the region $\mathcal{R}_{ij}$ is bounded as follows

$$\begin{aligned}
\int_{\mathcal{R}_{ij}} [f(x,y) - p_{ij}(x,y)]^2 &\le K_{ij}^2 \int_{\mathcal{R}_{ij}} \left((x - x_{ij})^2 + (y - y_{ij})^2 \right)^{\alpha} \\
&\le K_{ij}^2 \int_{\mathcal{R}_{ij}} (T\sqrt{2}2^{-J})^{2\alpha} \\
&\le K_{\max}^2 2^{\alpha} T^{2\alpha} 2^{-2\alpha J} T^2 2^{-2J} \\
&= K_0 2^{-2\alpha J} T^2 2^{-2J}; K_0 = K_{\max}^2 2^{\alpha} T^{2\alpha}. \quad (5.4)
\end{aligned}$$

where (x_{ij}, y_{ij}) is the origin coordinate of the quadtree node associated with region $\mathcal{R}_{ij}$ and $K_{\max} = \max_{0 \le i,j < 2^J} K_{ij}$.

The quadtree algorithm codes the polynomial $p_{ij}(x,y)$ by quantizing its $\frac{(P+1)(P+2)}{2}$ coefficients. Assume that the algorithm assigns r_{ij} bits to the polynomial $p_{ij}(x,y)$. Suppose that the polynomials $p_{ij}(x,y), 0 \le i,j < 2^J$, are bounded in magnitude by some constant A. Thus, the polynomial quantization distortion d_{ij} for the region $\mathcal{R}_{ij}$ due to the coefficient quantization can be bounded as follows

$$\begin{aligned}
d_{ij} &= \int_{\mathcal{R}_{ij}} \left(p_{ij}(x,y) - \widehat{p}_{ij}(x,y) \right)^2 \\
&\le \frac{1}{4} A^2 T^2 2^{-2J} (P+1)^2 (P+2)^2 2^{-\frac{4}{(P+1)(P+2)} r_{ij}}; \text{ from (4.21)} \\
&= K_1 2^{-2J} 2^{-\frac{4}{(P+1)(P+2)} r_{ij}}; \quad K_1 = \frac{1}{4} A^2 T^2 (P+1)^2 (P+2)^2. \quad (5.5)
\end{aligned}$$

Hence, the coding distortion D_{ij} for the region $\mathcal{R}_{ij}$ can be written as follows

$$\begin{aligned}
D_{ij} &= \int_{\mathcal{R}_{ij}} \left(f(x,y) - \widehat{p}_{ij}(x,y) \right)^2 \\
&= \int_{\mathcal{R}_{ij}} \left(f(x,y) - p_{ij}(x,y) + p_{ij}(x,y) - \widehat{p}_{ij}(x,y) \right)^2 \\
&= \int_{\mathcal{R}_{ij}} [f(x,y) - p_{ij}(x,y)]^2 + \int_{\mathcal{R}_{ij}} [p_{ij}(x,y) - \widehat{p}_{ij}(x,y)]^2, \text{ due to (5.3)} \\
&\le K_0 2^{-2\alpha J} T^2 2^{-2J} + K_1 2^{-2J} 2^{-\frac{4}{(P+1)(P+2)} r_{ij}}, \text{ from (5.4) and (5.5). (5.6)}
\end{aligned}$$

The quadtree scheme provides the piecewise polynomial approximation $p(x,y) = \sum_{i,j=0}^{2^J-1} p_{ij}(x,y)1_{\mathcal{R}_{ij}}(x,y)$, which is further quantized to $\widehat{p}(x,y) = \sum_{i,j=0}^{2^J-1} \widehat{p}_{ij}(x,y)1_{\mathcal{R}_{ij}}(x,y)$. Therefore, the overall distortion can be expressed as follows

$$\begin{aligned} D &= \int_{[0,T]^2} [f(x,y) - \widehat{p}(x,y)]^2 \\ &= \sum_{i,j=0}^{2^J-1} \int_{\mathcal{R}_{ij}} [f(x,y) - \widehat{p}_{ij}(x,y)]^2 \\ &\leq \sum_{i,j=0}^{2^J-1} \left[K_0 2^{-2\alpha J} T^2 2^{-2J} + K_1 2^{-2J} 2^{-\frac{4}{(P+1)(P+2)} r_{ij}} \right], \text{ from (5.6)} \\ &= K_0 T^2 2^{-2\alpha J} + \sum_{i,j=0}^{2^J-1} K_1 2^{-2J} 2^{-\frac{4}{(P+1)(P+2)} r_{ij}}. \end{aligned} \tag{5.7}$$

Again, the structure of (5.7) is such that R-D optimization forces all the leaves to have the same distortion.[2] That means, all the leaves will be allocated the same rate, i.e. $r_{ij} = r, \forall i,j = 0,\ldots,2^J-1$. Furthermore, for the given bit-budget R, the algorithm selects J and r such that the distortion of each leaf is of the order $O(2^{-2\alpha J - 2J})$.[3] Thus, the scheme allocates $r = \frac{(P+1)(P+2)}{2}\alpha J$ bits to each polynomial associated with a leaf. Therefore, the overall distortion can be bounded as follows

$$\begin{aligned} D &\leq \left(K_0 T^2 + K_1\right) 2^{-2\alpha J} \\ &= K_2 2^{-2\alpha J}; \quad K_2 = \left(K_0 T^2 + K_1\right) \\ &\leq K_2 \left(1 + \frac{\log 2J}{2J}\right)^{\alpha} 2^{-2\alpha J}. \end{aligned} \tag{5.8}$$

Since the full quadtree decomposition algorithm does not need to code the tree structure, the total bit rate is equal to the bits required for quantizing the polynomial coefficients associated with 2^{2J} quadtree leaves at the tree depth J. Hence, the net rate becomes

$$\begin{aligned} R &= 2^{2J} r = 2^{2J} \frac{(P+1)(P+2)}{2} \alpha J \\ R &= K_3 2J 2^{2J}; \quad K_3 = \frac{(P+1)(P+2)}{4}\alpha \end{aligned} \tag{5.9}$$

$$\log_2 \left(\frac{R}{K_3}\right) = \log_2 2J + 2J. \tag{5.10}$$

Combining (5.8) and (5.9) leads to the following asymptotic R-D behavior

$$D \leq d_3 \left(\frac{\log R}{R}\right)^{\alpha}, \tag{5.11}$$

[2] We obtain this result by using the usual reverse water filling technique. That is, $\frac{\partial D}{\partial r_{ij}} =$ constant.

[3] Otherwise, one of the terms in (5.6) will dominate the overall distortion and the R-D performance can be improved by readjusting J and r.

where $d_3 = K_2\left(K_3^2+1\right)^{\alpha}, K_2 = \left[K_{\max}^2 2^{\alpha} T^{2\alpha} + \frac{A^2(P+1)^2(P+2)^2}{4}\right] T^2$ and $K_3 = \frac{(P+1)(P+2)}{4}\alpha$.
Therefore, the full quadtree decomposition algorithm achieves the stated asymptotic R-D behavior. □

Corollary 5.1 *The prune quadtree coding algorithm achieves the following asymptotic R-D behavior*

$$D_{Prune}(R) \leq d_3 \left(\frac{\log R}{R}\right)^{\alpha}, \tag{5.12}$$

for 2-D smooth functions, which are uniformly Hölder α over $[0,T]^2$.

Proof: Since the prune quadtree algorithm results into the full quadtree algorithm, when no pruning is performed, the prune quadtree scheme will perform at least as well as the full tree scheme. Thus, the prune tree scheme will also achieve the R-D behavior given by (5.12). ◇

Corollary 5.2 *The prune-join quadtree algorithm achieves the following asymptotic R-D behavior*

$$D_{Prune\text{-}Join}(R) \leq d_3 \left(\frac{\log R}{R}\right)^{\alpha}, \tag{5.13}$$

for 2-D smooth functions, which are uniformly Hölder α over $[0,T]^2$.

Proof: Since the prune-join quadtree algorithm leads to the full quadtree algorithm, when neither pruning nor neighbor joining is performed, the prune-join tree scheme will perform at least as well as the full tree scheme. Hence, the prune-join tree scheme will also achieve the asymptotic R-D behavior indicated by (5.13). ◇

Therefore, we can conclude that the full, the prune and the prune-join schemes achieve the similar asymptotic R-D behavior $\left(d_3\left(\frac{\log R}{R}\right)^{\alpha}\right)$ for 2-D Hölder α smooth functions. However, note that the given R-D bound is a weak upper-bound for both the prune and the prune-join coding schemes.

Remark: It is worth noticing that the presented R-D analysis can be extended to the higher dimensional N-D case by using 2^N-tree segmentation method, where 2^N simply indicates the number of children of a node obtained by its decomposition. That means, the binary tree is 2^1-tree, the quadtree is 2^2-tree, the oct-tree is 2^3-tree, and so on. By following the same logic of Theorem 5.1, one can show that, for N-D uniformly Hölder α smooth functions, the full 2^N-tree decomposition algorithm, which codes every tree node at the tree depth by a N-D polynomial approximation, achieves the following asymptotic R-D behavior

$$D_{\text{Full}}(R) \leq d_4 \left(\frac{\log R}{R}\right)^{\frac{2\alpha}{N}}, \tag{5.14}$$

for some positive constant d_4 which depends on the smoothness, magnitude and region of support of the function.

We can notice that, by putting $N = 1$ and $N = 2$ in (5.14), we obtain the R-D behaviors derived for the binary tree and quadtree algorithms, respectively.

5.3 R-D Analysis for 2-D Piecewise Smooth Functions

Similar to the 1-D case, we can define a 2-D piecewise smooth function $f(x, y)$, $(x, y) \in [0, T]^2$ with M smooth pieces, in the following way

$$f(x, y) = \sum_{i=1}^{M} f_i(x, y)\mathbf{1}_{\mathcal{R}_i}(x, y), \tag{5.15}$$

where $\bigcup_{i=1}^{M} \mathcal{R}_i = [0, T]^2$ and $f_i(x, y)$, the function associated with the region $\mathcal{R}_i$, is uniformly Hölder α over $[0, T]^2$. Furthermore, region boundaries are also 1-D piecewise smooth curves whose pieces are uniformly Hölder α over $[0, T]$. Since the class of 2-D piecewise smooth signals is much more general than the class of piecewise polynomial images, it can better model real images encountered in practice. That is why we will study the R-D behaviors of the quadtree algorithms for piecewise smooth functions in this section.

For the 1-D case, it has been shown in [21] that a 1-D piecewise smooth function, whose pieces are Hölder α-smooth, can be expressed as a sum of a 1-D piecewise polynomial with pieces of maximum degree $P = \lfloor \alpha \rfloor$ and a residual function which is Hölder α-smooth. However, we show in the following theorem that such a decomposition cannot be performed in 2-D.

Theorem 5.2 *A 2-D piecewise smooth function $f(x, y)$, whose pieces are uniformly Hölder α-smooth, cannot be written as a sum of a 2-D piecewise polynomial $p(x, y)$ with finite number of pieces of finite degree and a residual function $r(x, y)$ which is uniformly Hölder α-smooth.*

Proof: We prove this theorem by contradiction. Let us consider a very simple piecewise smooth function, which is composed of only two 2-D α-smooth functions separated by a smooth boundary. Assume that there exists a 2-D piecewise polynomial with pieces of finite degrees, which make the residual function α-smooth. This implies that the 2-D piecewise polynomial representation completely kills the singularity along the smooth boundary. Hence, the polynomials need to eliminate the singularities at infinitely many points on the smooth boundary. This means that they need to satisfy infinite number of constraints. However, we know that no polynomial of finite degree can satisfy infinitely many constraints. Therefore, we cannot find a finite degree 2-D piecewise polynomial representation, which completely kills the singularity along the smooth boundary. Thus, a 2-D piecewise smooth function $f(x, y)$, whose pieces are Hölder

α-smooth, cannot be expressed as a sum of a 2-D piecewise polynomial $p(x, y)$ with finite number of pieces of finite degree and a residual function $r(x, y)$ which is Hölder α-smooth. □

Moreover, this results holds even if the boundary is a 1-D polynomial function. This is because still 2-D polynomials have to satisfy infinite number of constraints to eliminate singularities at infinitely many points on the polynomial boundary. Again, since no polynomial of finite degree can satisfy infinitely many constraints, we cannot find a finite degree 2-D piecewise polynomial representation, which completely kills the singularity along the polynomial boundary.

Remark: An important practical implication of Theorem 5.2 is that a two stage compression system, piecewise polynomial approximation followed by a residual coder, cannot improve the overall R-D performance compared to simple piecewise polynomial approximation. This is because the residual function, despite having low energy, is not smooth and is geometrically more complex which makes it difficult to compress efficiently.

5.3.1 R-D Analysis for the Horizon Model

For the sake of simplicity, we carry out the analysis for a simpler image model known as the Horizon model [16]. An image $f(x, y)$ of the Horizon model is defined on the unit square $[0, 1]^2$ in the following way:

$$f(x, y) = 1_{y \geq b(x)}, \qquad 0 \leq x, y \leq 1, \tag{5.16}$$

where the Horizon boundary function $b(x)$ is uniformly Hölder α, and $b(x)$ has finite length inside the unit square.

Oracle R-D behavior

Since the Horizon model image is white above the boundary and black below the boundary, it has the complexity of a 1-D function. Therefore, when coding such an image, an oracle based method could simply spend all the available bit rate to code the smooth edge. Given a bit budget R, coding the edge b produces $\hat{b}$ and the corresponding 2-D function $\hat{f}$. The distortion can be written as follows

$$\begin{aligned} D(R, f) &= \int_{[0,1]^2} (f - \hat{f})^2 \leq \int_{[0,1]} |b - \hat{b}| \\ &\leq \sqrt{\int_{[0,1]} (b - \hat{b})^2} = \sqrt{D(R, b)}, \end{aligned}$$

where the second inequality comes from the Schwartz's inequality. Since the edge $b(x)$ is uniformly Hölder α smooth, coding it using a wavelet basis with enough vanishing moment gives $D(R, b) \sim R^{-2\alpha}$ [9]. Thus, the oracle R-D behavior for the Horizon model is as follows

$$D(R, f) \sim R^{-\alpha}. \tag{5.17}$$

R-D performance of the prune quadtree algorithm

Theorem 5.3 *For the Horizon model with uniformly Hölder α smooth boundary, the prune quadtree coding algorithm achieves the following asymptotic R-D behavior*

$$D_{Prune}(R) \sim d_5 \left(\frac{\log R}{R}\right)^{\alpha}, \tag{5.18}$$

where d_5 is a positive constant which depends on the smoothness, magnitude and length of the boundary function.

Proof: Assume that the Horizon model image has an edge of finite length L_{Edge}. Suppose that the quadtree decomposition is performed up to the depth J. Any tree node either contains the edge segment or can be represented by a smooth region (black or white). The quadtree nodes without edge will be simply assigned 1 bit to indicate if it is black or white. This ensures that smooth nodes are represented losslessly (zero distortion) just using 1 bit. However, an edge block is approximated by an edge tile which consists of two constant regions separated by a polynomial approximation of the edge segment. That means, at high rates, edge blocks need to be further divided to capture the edge and to improve the R-D performance. Therefore, at high rates, our quadtree algorithm recursively divides the blocks with edges only. Let n_j be the number of dyadic squares at tree level j that intersect with edge curve on the unit square. Since the edge has finite length L_{Edge}, we have

$$n_j \quad \sim \quad L_{\text{Edge}} 2^j. \tag{5.19}$$

Since only edge nodes are divided further, the total number N_{All} of nodes can be counted as the number of children of all dyadic edge squares up to level $J-1$ that intersect with the boundary $b(x)$. That means,

$$\begin{aligned} N_{\text{All}} &\leq 4 \sum_{j=0}^{J-1} n_j \\ N_{\text{All}} &\sim 4 L_{\text{Edge}} 2^J, \text{ using (5.19).} \end{aligned} \tag{5.20}$$

Clearly, at high rates, all the edge leaves will lie at the tree-depth J in the worst case scenario. Hence, the number N_{Edge} of edge leaves can be bounded as follows: $N_{\text{Edge}} \leq n_J \sim L_{\text{Edge}} 2^J$. Thus, the number of leaves to be coded grows exponentially with respect to the depth J. To code an edge leaf i, the tree scheme approximates its edge segment by a polynomial p_i of degree $P = \lfloor \alpha \rfloor$ and code the polynomial by quantizing its coefficients.[4] If we allocate r_i bits to the edge leaf i of side-size 2^{-J}, the associated distortion D_i can be bounded as

[4] The scheme also uses 1 bit to indicate the value (black or white) of the region above the polynomial.

follows

$$\begin{aligned} D_i &= \int |b(x) - \widehat{p}_i(x)| \\ &\leq \sqrt{\int |b(x) - \widehat{p}_i(x)|^2 \cdot 2^{-J}}, \text{ from Schwartz's inequality} \\ &\leq \sqrt{\left(K_0 2^{-2\alpha J} 2^{-J} + A_0^2 2^{-J} 2^{-\frac{2}{P+1} r_i}\right) 2^{-J}}, \text{ using (3.7)} \\ &\leq \sqrt{K_0} 2^{-\alpha J} 2^{-J} + A_0 2^{-J} 2^{-\frac{1}{P+1} r_i}. \end{aligned} \tag{5.21}$$

The total distortion is the sum of the distortions of all tree leaves. Since leaves without edges are coded losslessly, the net distortion becomes

$$\begin{aligned} D &= \sum_{i=1}^{N_{\text{Edge}}} D_i \\ &\leq N_{\text{Edge}} \sqrt{K_0} 2^{-\alpha J} 2^{-J} + \sum_{i=1}^{N_{\text{Edge}}} A_0 2^{-J} 2^{-\frac{1}{P+1} r_i}. \end{aligned} \tag{5.22}$$

Clearly, R-D optimization will force all the edge leaves to have the equal distortion. That means, all the edge leaves will be allocated the same rate, i.e. $r_i = r, i = 1, \ldots, N_{\text{Edge}}$. Moreover, for the given bit-budget R, the algorithm selects J and r such that the distortion of each edge leaf is of the order $O(2^{-\alpha J - J})$.[5] Hence, the coding scheme allocates $r = (P+1)\alpha J$ bits to each polynomial associated with a leaf. Therefore, the overall distortion can be bounded as follows

$$\begin{aligned} D &\leq N_{\text{Edge}} \left(\sqrt{K_0} + A_0\right) 2^{-\alpha J} 2^{-J} \\ &\sim L_{\text{Edge}} \left(\sqrt{K_0} + A_0\right) 2^{-\alpha J}; \text{ as } N_{\text{Edge}} \sim L_{\text{Edge}} 2^J \\ &\sim K_1 2^{-\alpha J}; \qquad K_1 = L_{\text{Edge}} \left(\sqrt{K_0} + A_0\right). \end{aligned} \tag{5.23}$$

The total bitrate is the sum of the costs of coding the tree structure and the quantized model parameters of the smooth and edge leaves. The bitrate R_{Tree} required to code the tree structure is equal to the total number N_{All} of nodes in the pruned tree. Thus, (5.20) gives $R_{\text{Tree}} \sim 4L_{\text{Edge}} 2^J$. Since a smooth leaf is coded using only 1 bit, the bitrate $R_{\text{SmoothLeaves}}$ needed for smooth leaves is bounded by N_{All}. That is, $R_{\text{SmoothLeaves}} \sim 4L_{\text{Edge}} 2^J$. To code the edge leaves, the algorithm uses $R_{\text{EdgeLeaves}} = N_{\text{Edge}}((P+1)\alpha J + 1)$ bits.[6] Hence, the total

[5] Otherwise, one of the terms in (5.21) will dominate the overall distortion and the R-D performance can be improved by reselecting J and r.

[6] Recall that the scheme uses 1 bit to code the value (black or white) of the region above the polynomial.

bitrate can be computed as follows

$$\begin{aligned} R &= R_{\text{Tree}} + R_{\text{SmoothLeaves}} + R_{\text{EdgeLeaves}} \\ &\sim 4L_{\text{Edge}}2^J + 4L_{\text{Edge}}2^J + L_{\text{Edge}}2^J((P+1)\alpha J + 1) \\ &\sim L_{\text{Edge}}(P+1)(\alpha+9)J2^J; \text{ as at high rates } J > 1 \\ &\sim K_2 J 2^J; \qquad K_2 = L_{\text{Edge}}(P+1)(\alpha+9). \end{aligned} \tag{5.24}$$

Combining (5.23) and (5.24) provides the following asymptotic R-D behavior

$$D \sim d_5 \left(\frac{\log R}{R}\right)^{\alpha}, \tag{5.25}$$

where $d_5 = K_1 K_2^{\alpha} = L_{\text{Edge}} \left(\sqrt{K_0} + A_0\right) \left(L_{\text{Edge}}(P+1)(\alpha+9)\right)^{\alpha}$.
Therefore, the prune quadtree algorithm achieves the announced R-D performance. □

Corollary 5.3 *The prune-join quadtree algorithm achieves the following asymptotic R-D behavior*

$$D_{Prune\text{-}Join}(R) \sim d_5 \left(\frac{\log R}{R}\right)^{\alpha}, \tag{5.26}$$

for the Horizon model with uniformly Hölder α smooth boundary.

Proof: Since the prune-join quadtree algorithm results in the prune quadtree algorithm, when no neighbor joining is performed, the prune-join tree scheme will perform at least as well as the prune tree scheme. Hence, the prune-join tree scheme will also achieve the asymptotic R-D behavior indicated by (5.26). ◇

Therefore, for the Horizon model, both the prune and prune-join tree algorithms achieve the near-optimal asymptotic R-D decay $\left(d_5 \left(\frac{\log R}{R}\right)^{\alpha}\right)$. The suboptimality factor $(\log R)^{\alpha}$ is clearly a penalty for using a tree structure.

Furthermore, by combining the coding strategy for the Horizon model with that of the 1-D piecewise smooth signals given in Section 3.3.2, it is straightforward to show that this result will also hold when the edge curve $b(x)$ is piecewise smooth.[7] The key point is that the tree nodes with singular points of the edge curve cannot be well approximated by edge tiles. So these nodes will be decomposed deeper in comparison to the nodes which contain edge segment without any singular point of the edge curve. Our earlier analysis has shown that R-D optimization forces all the tree leaves to have the distortion of the same order. Suppose that the tree nodes with singular points of the edge curve are decomposed up to level J, so their distortions are of the order $O\left(2^{-2J}\right)$. Similarly, if the edge-nodes, which contain an edge segment without any singular point of the edge curve, are decomposed up to level J_1, then their distortions are of

[7] We have already shown in Section 3.3.2 that the binary tree algorithms achieve the correct R-D behavior for 1-D piecewise smooth signals.

the order $O\left(2^{-(\alpha+1)J_1}\right)$ from (5.21). The similar distortion constraint leads to the following choice of decomposition levels: $J = \frac{\alpha+1}{2}J_1$. The above choice of decomposition levels along with the bit allocation strategy given for the Horizon model in Theorem 5.3 leads to the polynomially decaying R-D behavior for binary images with piecewise smooth edge curve.

Now, let us consider an image with two 2-D Hölder α smooth functions separated by a Hölder α smooth boundary. Due to Corollary 5.1 and Theorem 5.3, it is clear that the prune tree algorithm can efficiently code both the smooth regions as well as the smooth boundary. Thus, the prune tree scheme will achieve the following asymptotic R-D behavior

$$D_{\text{Prune}}(R) \sim d_6 \left(\frac{\log R}{R}\right)^{\alpha}, \tag{5.27}$$

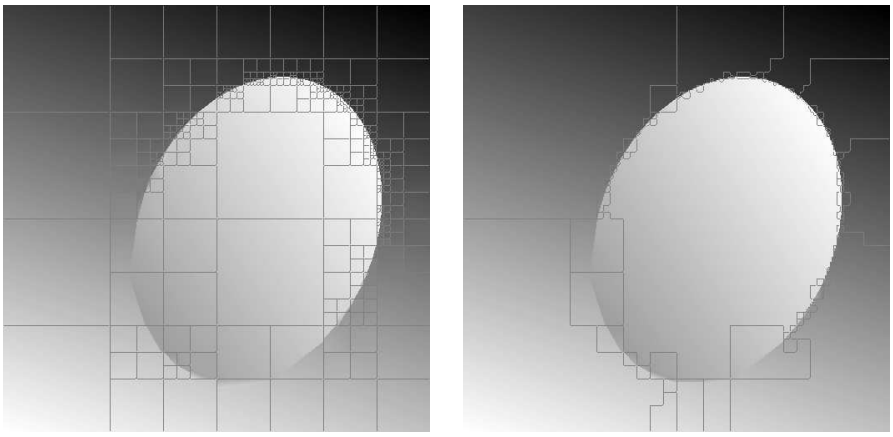
where d_6 is a positive constant which depends on the smoothness, magnitude and region of support of the given image.

Remark: For piecewise smooth images with piecewise smooth boundaries, the tree algorithm has to efficiently deal with three different type of nodes, that is, nodes with smooth regions, nodes with two smooth regions separated by a smooth boundary, and nodes with a singular point of a piecewise smooth boundary. Assume that the nodes with two smooth regions separated by a smooth boundary and nodes with smooth regions are decomposed up to levels J_1 and J_2, respectively. On the other hand, tree nodes with singular points of the edge curve will be decomposed deeper in comparison to the other type of nodes. Suppose that the tree nodes with singular points of the edge curve are decomposed up to level J. Again, the similar distortion constraint leads to the following choice of decomposition levels: $J = \frac{\alpha+1}{2}J_1 = (\alpha+1)J_2$. The above choice of decomposition levels along with the bit allocation strategies given in Theorem 5.1 and Theorem 5.3 results into the polynomially decaying R-D behavior for piecewise smooth images with piecewise smooth boundaries.

5.4 Simulation Results

To verify the theoretical results, we perform numerical simulations for the Horizon class, where a smooth edge separates two smooth regions. In particular, we select images with elliptical edges. Figure 5.1 shows the partitioning performed by the quadtree schemes for the image with an elliptical edge. As expected, the tree algorithms perform finer segmentation along the elliptical edge, while smooth regions are represented by larger blocks. It is self evident from Figure 5.2 that the wavelet based coder (JPEG2000) fails to capture the regularity of the smooth elliptical edge. Thus, it is bound to perform sub-optimally. Finally, we compare the R-D performance of the quadtree algorithms with the wavelet based coder (JPEG2000) in Figure 5.3, which clearly shows the superiority of our tree algorithms over JPEG2000.

Figure 5.4 shows a phantom image (which is composed of several elliptical edges) along with an encoded image obtained by JPEG2000. Figure 5.5 displays the encoded phantom images obtained by the prune and prune-join quadtree algorithms. Figures 5.4 and 5.5 clearly show that the quadtree algorithms achieve better objective (PSNR) as well as subjective (visual) quality compared to JPEG2000. In particular, the prune-join quadtree algorithm achieves the best R-D performance at low rates.



(a) Prune tree segmentation (Rate=.03 bpp, PSNR=44.43 dB).

(b) Prune-join tree segmentation (Rate=.02 bpp, PSNR=44.24 dB).

Figure 5.1: Segmentation performed by the quadtree algorithms for a piecewise linear image with an elliptical edge.

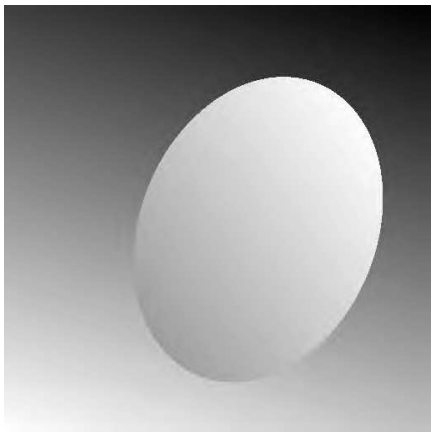


Figure 5.2: Reconstructed image by JPEG2000 (Rate=.065 bpp, PSNR= 43.81 dB).

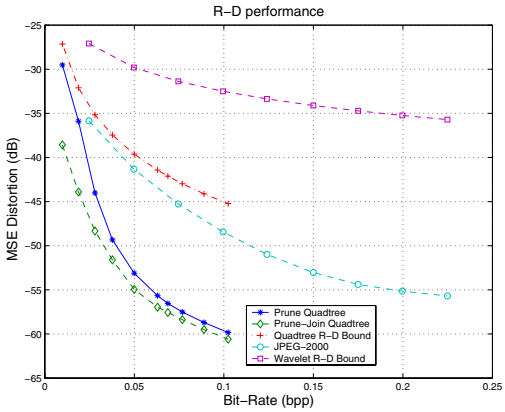
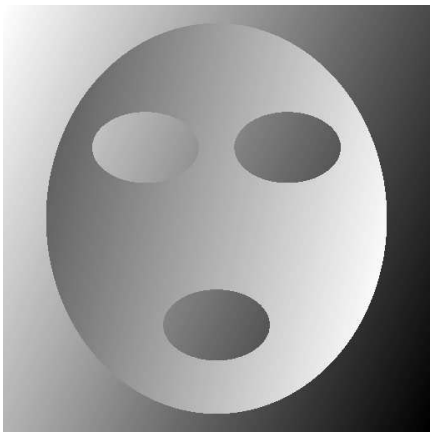


Figure 5.3: R-D performance comparison of the proposed quadtree algorithms and JPEG2000 for images with smooth elliptical edges.

5.5 Conclusions

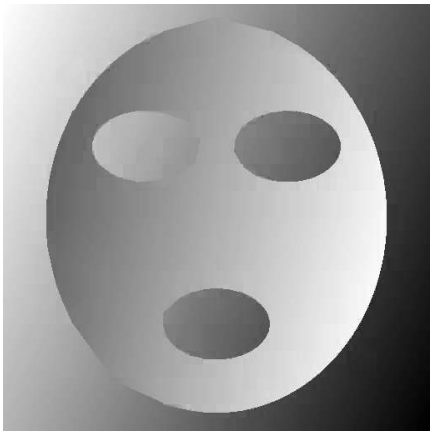
This chapter shows that both the prune and prune-join tree algorithms achieve the near-optimal asymptotic R-D behavior $\left(D(R) \sim d_3 \left(\frac{\log R}{R}\right)^{\alpha}\right)$ for 2-D smooth functions. Note that the suboptimality factor $(\log R)^{\alpha}$ can be seen as a penalty for using a tree structure. We have also presented a strong negative result as Theorem 5.2, which states that a 2-D piecewise smooth function $f(x, y)$ cannot be decomposed as a sum of a 2-D piecewise polynomial $p(x, y)$ and a smooth residual function $r(x, y)$. However, the proposed tree algorithms still achieve a polynomially decaying R-D behavior (Theorem 5.3 and (5.27)) for 2-D piecewise smooth functions with smooth boundaries. Simulation results shown in Figure 5.3 also confirm the derived R-D behaviors of the tree schemes. Therefore, the theoretical as well as numerical results show that the tree based schemes can efficiently represent piecewise smooth images, which in turn can efficiently model real life images. In the next chapter, we will explore new applications, namely image denoising and stereo image coding, for the proposed tree based schemes.



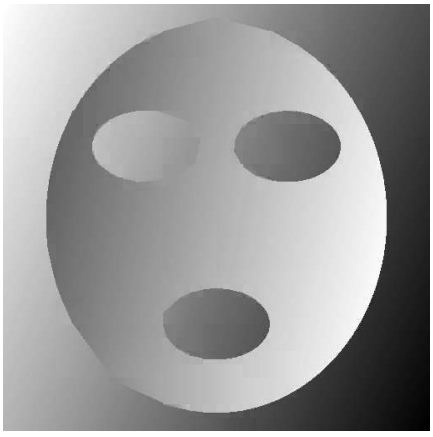
(a) Original phantom image.

(b) JPEG2000 (Rate=.09 bpp, PSNR=38.25 dB).

Figure 5.4: Original phantom image along with an encoded image obtained by JPEG2000.



(a) Prune tree (Rate=.065 bpp, PSNR=39.01 dB).



(b) Prune-join tree (Rate=.04 bpp, PSNR=39.06 dB).

Figure 5.5: Encoded phantom images obtained by the prune and prune-join quadtree algorithms.

Chapter 6

New Applications of Tree Algorithms: Denoising and Stereo Image Coding

6.1 Introduction

In the previous chapters, we have presented tree segmentation based algorithms and analyzed their compression performance for piecewise smooth signals/images. In particular, we have shown that the proposed tree based schemes can not only accurately capture singularities but also efficiently represent the regular structure separated by these singularities. Since singularities, like edges in real images, represent one of the most important perceptual information in an image, their precise modeling is very critical in several signal/image processing applications, like denoising and computer vision. As we know that the proposed tree algorithms are efficient in exploiting the regularity of singularities as well as that of the associated smooth regions, these algorithms may show some potential for image processing applications requiring efficient representation of singularities.

In this chapter, we focus on two image processing applications, namely denoising and stereo image compression, which require to efficiently exploit the regularity of both the singularities and smooth pieces separated by the singularities. In the next section, we consider the problem of denoising and present a solution using our tree based algorithm. The key is to treat the denoising problem as a compression problem at low rates. The intuition is that, at low rates, the coding scheme captures the dominant (smooth and geometrical) features only, which basically belong to the original signal. We then present simulation results and compare our compression based denoising scheme with state of the art wavelet based denoising scheme in Section 6.2.1. In Section 6.3, we address

the problem of stereo image compression and propose a novel rate-distortion (R-D) optimized disparity based coding scheme for stereo images. In Section 6.3.2, we present simulation results for the proposed scheme and compare these results with the performance of a fixed block size disparity estimation/disparity compensation (DE/DC) based JPEG2000 stereo image coder. Finally, we conclude with a discussion of further research directions in Section 6.4.

6.2 Denoising Problem

Denoising is a classical problem in estimation theory, which has received an extensive treatment in the literature [6, 11, 18, 16, 21, 24, 39, 69, 75]. In the denoising problem, the goal is to estimate the original signal x from an observed *noisy* signal y. More formally, consider that the signal x has been corrupted by the additive white noise z. Thus, the observed signal is

$$y = x + z.$$

Our goal is to obtain an estimate, $\widehat{x}$, of x from y such that the mean squared error $E\|x - \widehat{x}\|^2$ is minimized. Theoretically, the best estimate is

$$\widehat{x} = E\left[x|y\right], \tag{6.1}$$

which is quite difficult to solve in general. This is because Equation (6.1) is nonlinear, and the conditional probability density $p_{x|y}$ required for solving (6.1) is difficult to calculate. Therefore, one generally settles for the best linear estimate. And this can be obtained by the Wiener filtering approach.[1] However, the Wiener filtering method requires information about the spectra of the noise and the original signal and it works well only if the underlying signal is smooth. When the given signal is piecewise smooth, then block based Wiener filtering can be used. But this approach generally fails to perform well as fixed segmentation cannot capture the real singularities of the underlying piecewise smooth signal. To overcome the weakness of Wiener filtering, Donoho and Johnstone proposed the wavelet based denoising scheme in [18]. This wavelet based denoising scheme basically projects the noisy signal on the subspace, where the original signal is supposed to live. This scheme assumes that the noise is *low power and white* phenomenon and works as follows

1. Apply the wavelet transform on the noisy signal.
2. Estimate noise strength σ from the high frequency sub-bands.
3. Compute the appropriate threshold t_0, e.g., $t_0 = 3\sigma$.
4. Threshold the wavelet coefficients via hard or soft thresholding scheme.
5. Reconstruct the signal from the thresholded wavelet coefficients. This reconstructed signal is our denoised signal.

[1] Wiener filter, that is a linear estimator, is optimal if and only if x and z are jointly Gaussian processes.

This scheme generally works well but fails to perform the segmentation according to the real singularities of the given piecewise smooth signal. It has been shown in [9, 40] that the wavelet based coders perform suboptimally due to their failure to precisely model singularities. This suggests that the above wavelet based denoising scheme might have some limitations for the denoising application. That means, a good denoising scheme should be capable of performing the segmentation according to the real singularities of the underlying signal. Since the proposed prune-join tree algorithm accurately models both singularities and smooth pieces, we expect it to play an important role in denoising applications. Now, the key question is how to design a tree based denoising algorithm.

In [35], Natarajan has provided a crucial insight that the lossy compression of a noisy signal may lead to a filtered/denoised signal. The reason is as follows: At low rates, the coding scheme captures the smooth features only, which basically belong to the original signal. However, at high rates, coding scheme also tracks high frequency noise-like features. That means, as the algorithm codes the noisy signal from low rates to high rates, the method first tracks the signal up to a certain rate R_0. But above this rate R_0, the scheme starts tracking more noise than the signal. Moreover, the best denoising performance is obtained for the R-D operating point, which has the distortion equal to the noise strength. Figure 6.1 presents a typical R-D curve for a noisy signal. This graph clearly shows that the distortion reduces rapidly upto certain rate R_0. But beyond this rate, even a small reduction in distortion requires a large increase in rate as the coding scheme starts to track noise. This transition point R_0 is indeed the desired operating point for the coding scheme for the given signal. This point is called the knee point of the R-D curve, i.e., the point at which its second derivative attains a maximum.

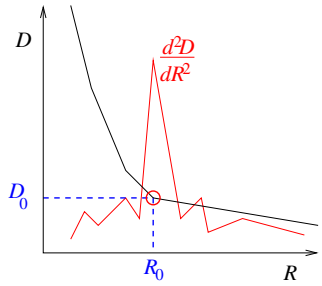
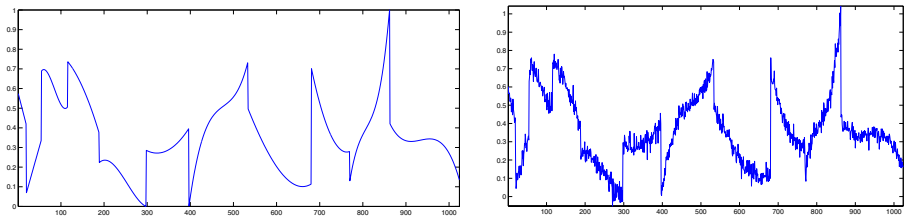


Figure 6.1: R-D curve and its second derivative for a noisy signal. (R_0, D_0) represents the knee point of the R-D curve.

Natarajan's insight suggests us to select a coding scheme, which can represent the underlying original signal more efficiently. Therefore, the knowledge about the class of the original signal can be used to achieve better denoising performance. Suppose that the problem of our interest is as follows: Consider a

piecewise polynomial signal shown in Figure 6.2, which has been corrupted by white Gaussian noise. What is a good way to recover the underlying piecewise polynomial from the noisy signal?



(a) Original piecewise cubic signal. (b) Noisy piecewise cubic signal.

Figure 6.2: Original and noisy piecewise cubic signals.

Since a piecewise polynomial signal can be precisely described by the singular-points and the polynomial models associated with segments, the key task is to correctly compute the segment boundaries and associated polynomial models. Our answer to the above problem is the prune-join tree algorithm as it can efficiently model both singularities and smooth polynomials. The tree based denoising scheme can be outlined as follows

Algorithm 6.1 ***The tree based denoising algorithm***

1. Compute the R-D curve using the tree coding scheme for the given noisy signal.
2. Estimate the knee point of the R-D curve by calculating the point where the second derivative of the R-D function achieves its maximum. Note that we approximate the second derivative by the second order difference.
3. Compute the coded signal representation for this R-D point.
4. This coded signal is our filtered/denoised signal.

In the next section, we perform denoising related experiments using the above described algorithmic framework.

6.2.1 Simulation Results: Denoising

1-D Scenario

In the 1-D case, we use the binary tree algorithms, proposed in Section 2.2, for denoising of signals. These algorithms basically employ polynomial models to characterize the underlying regular structure of signals. For denoising experimentation, we consider piecewise polynomial signals and piecewise smooth signals constructed from the lines of the lena image. Figures 6.3 and 6.4 show

the denoising performance of the proposed binary tree algorithms and wavelet based schemes for a noisy piecewise cubic signal. We can clearly observe that our tree based algorithms outperforms the classical hard thresholding based wavelet algorithm [18] and cycle-spinning based wavelet scheme [11]. In Figure 6.5, we compare the performance of our denoising algorithms against cycle-spinning based wavelet scheme [11]. The next signal of interest is a piecewise smooth signal constructed from the lines of the lena image. Figure 6.6 clearly shows that the tree based denoising schemes are competitive with the cycle-spinning based wavelet denoising scheme for piecewise smooth signals as well. Overall, these simulation results indicate the potential of the tree based denoising schemes.

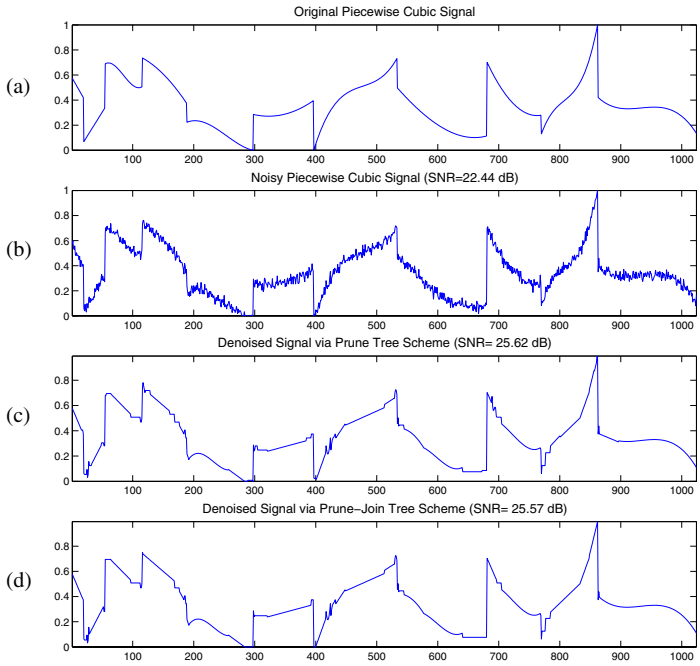


Figure 6.3: Denoising of a noisy piecewise cubic signal using the binary tree schemes. (a) Original signal. (b) Noisy signal: SNR= 22.44 dB. (c) Prune tree scheme: SNR= 25.62 dB. (d) Prune-join tree scheme: SNR= 25.57 dB.

2-D Scenario

In the 2-D case, the quadtree algorithms, which are described in Section 4.1, are employed for image denoising application. Since the quadtree algorithms use the linear-edge model explicitly in the approximation tile, they are well suited to capture the geometry hidden in noisy images. In this experimentation, we consider piecewise polynomial and real images. Figure 6.7 shows a piecewise quadratic image along with its noisy version. In Figure 6.8, we present the denoised images obtained by the prune and prune-join quadtree algorithms. On

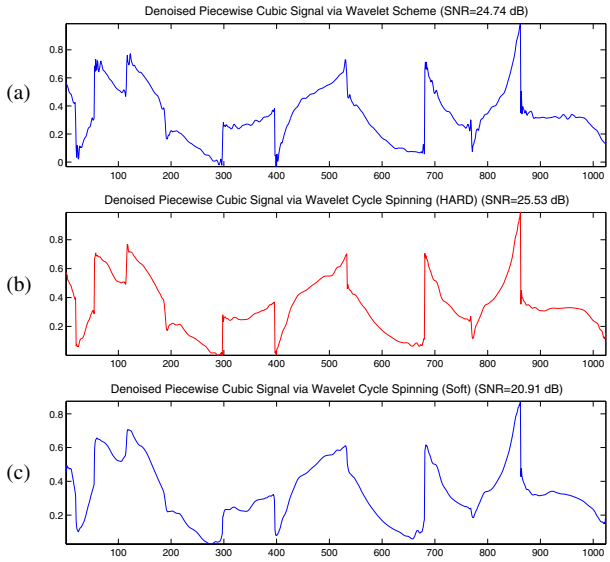


Figure 6.4: Denoising of a noisy piecewise cubic signal using the wavelet based schemes. (a) Standard wavelet scheme: SNR= 24.74 dB. (b) Cycle-spinning based wavelet scheme with hard thresholding: SNR= 25.53 dB. (c) Cycle-spinning based wavelet scheme with soft thresholding: SNR= 20.91 dB.

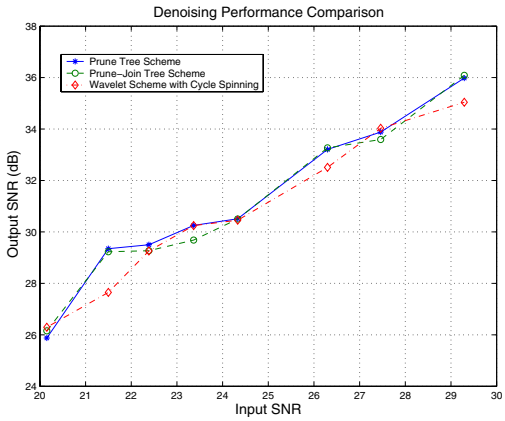
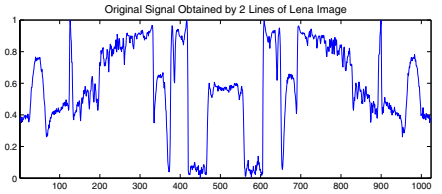


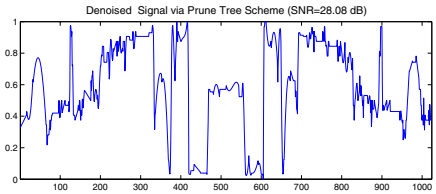
Figure 6.5: SNR performance comparison of the binary tree and wavelet based schemes for the piecewise cubic signal.

the other hand, Figure 6.9 displays the denoised images provided by the standard un-decimated wavelet transform based denoising method and the adaptive directional wavelet transform based denoising scheme [69].

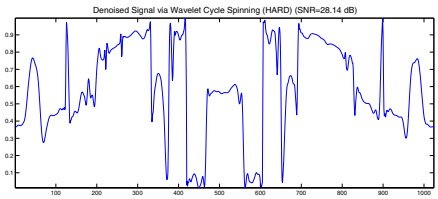
In Figure 6.10, we show the cameraman image and its noisy version with SNR= 17.72 dB. Figure 6.11 presents the denoised images obtained by the prune and prune-join quadtree schemes. Figure 6.12 shows the denoised images



(a) Original and noisy signals. For noisy signal: SNR= 26.8 dB.



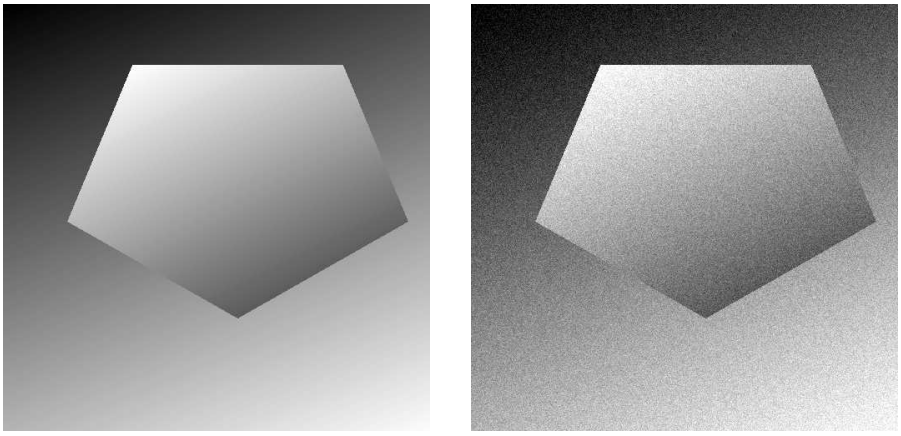
(b) Binary tree based denoising. Prune tree scheme: SNR= 28.08 dB. Prune-join tree scheme: SNR= 28.02 dB.



(c) Wavelet based denoising: SNR= 28.14 dB.

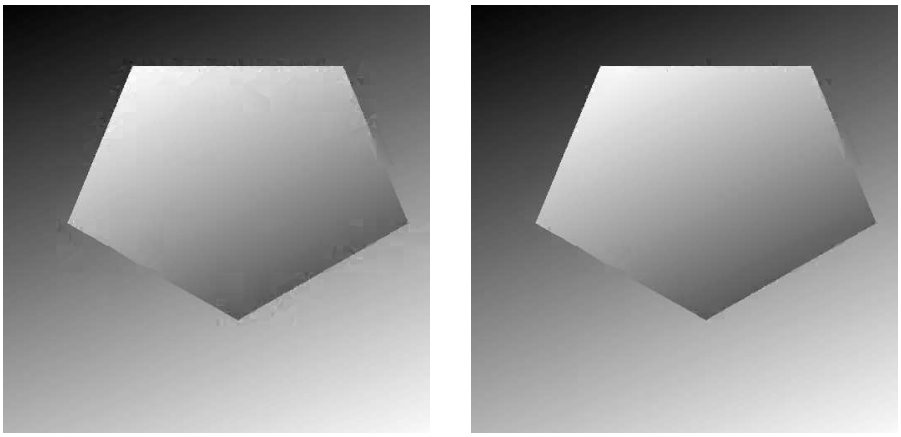
Figure 6.6: Denoising of a noisy signal, constructed from lines of the lena image, using the binary tree and wavelet based schemes.

provided by the standard un-decimated wavelet transform and the adaptive directional wavelet transform based denoising schemes. One can easily see that the tree algorithms preserve the sharpness of edges while the wavelet based schemes suffer from the ringing artefacts around the edges. These simulation results clearly demonstrate that the proposed tree based schemes perform better than the wavelet based schemes if the original image exactly fits the piecewise polynomial model. Even for real images, the performance of the tree based denoising schemes is competitive with that of the current state of the art wavelet based denoising schemes.



(a) Original image. (b) Noisy image.

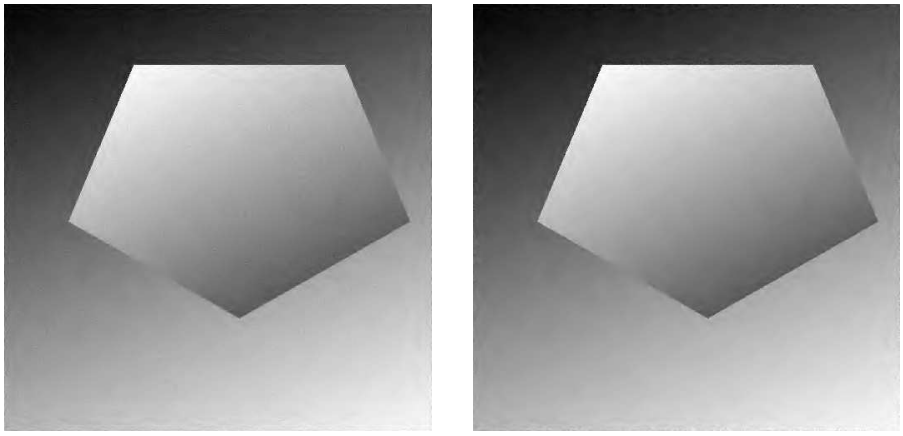
Figure 6.7: Original and noisy piecewise quadratic image. For noisy piecewise quadratic image, SNR= 20.32 dB.



(a) Prune quadtree based denoising (SNR= 36.91 dB).

(b) Prune-join quadtree based denoising (SNR= 37.96 dB).

Figure 6.8: Denoised images obtained by the prune and prune-join quadtree schemes.



(a) Standard un-decimated wavelet transform (SNR= 34.66 dB).

(b) Adaptive directional wavelet transform (SNR= 36.09 dB).

Figure 6.9: Denoised images provided by the standard un-decimated wavelet transform and adaptive directional wavelet transform based schemes.

(a) Original cameraman.

(b) Noisy cameraman.

Figure 6.10: Original and noisy cameraman image. For noisy cameraman image, SNR= 17.72 dB.

(a) Prune tree based denoising (SNR= 24.74 dB).

(b) Prune-join tree based denoising (SNR= 24.73 dB).

Figure 6.11: Denoised images obtained by the prune and prune-join quadtree schemes.

(a) Standard un-decimated wavelet transform (SNR= 24.42 dB).

(b) Adaptive directional wavelet transform (SNR= 24.82 dB).

Figure 6.12: Denoised images provided by the standard un-decimated wavelet transform and adaptive directional wavelet transform based schemes.

6.3 Stereo Image Compression

It is well known that the binocular or disparity information extracted from stereo images plays a crucial role in several fields like computer vision, remote sensing/handling, tele-presence style video conferencing, tele-medicine and 3-D cinema. In particular, the majority of multi-view visual coding schemes are based on disparity-field estimation and compensation. Moreover, the recent increase in stereo visual applications translates into a growing demand for efficient methods for the transmission and storage of stereo image/video pairs. If the stereo images are compressed and transmitted independently using the standard coding schemes, then the required bandwidth would need to be doubled. However, due to the binocular dependency between the stereo images, they contain significant redundant information in the form of inter-frame redundancy [38]. Thus, by exploiting this binocular redundancy in addition to the intra-frame redundancy present in the two constituent images, we can achieve significant data compression without sacrificing the overall image quality.

Coding of stereoscopic images is generally based on exploiting the correlation between the left and right images. This is achieved by computing a disparity field between the stereo image pair [31, 38, 42]. The disparity field represents the amount of shift one needs to perform on the pixels within one image (target) to find the corresponding pixels in the other image (reference). A popular approach for computing the disparity field is to partition the target image into a set of blocks and perform a block matching algorithm to find the best match in the reference image. For coding applications, this approach is known as disparity estimation/disparity compensation (DE/DC) due to its resemblance to motion estimation/motion compensation (ME/MC) methods, which are popular for video coding. The key concept of stereo image compression based on DE/DC is to use one of the images in the stereo pair as a reference and to predict the other image (the target). In Figure 6.13, we show a generic conditional coder/decoder structure for stereo image coding.

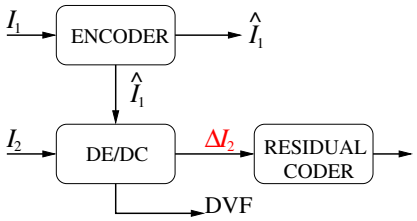


Figure 6.13: The conditional coder/decoder structure proposed by Perkins [38] for stereo image coding.

Disparity compensation based prediction for stereo images was first introduced by Lukacs [31]. In [38], Perkins proposed the conditional coder/decoder structure for the stereo image coding and analyzed the R-D behavior of this

structure. Aydinoglu *et al.* presented a coding scheme which combines disparity compensation and transform coding of the residual image in the single framework using an adaptive incomplete transform [1]. Hierarchical block matching algorithms have been used to generate a more homogeneous disparity fields that lead to better coding efficiency [49, 63].

In [4], a novel scheme is presented for the efficient exploitation of the zerotree algorithm in stereo image coding applications. Disparity field estimation based on low level features, such as edges, and on model/object based methods have also been used [58]. In [63], a feature/object based DE/DC scheme is proposed. This scheme determines a set of objects/features in both images and seeks correspondence between the two sets. Jiang [25] proposed a hybrid approach between block and object based schemes for the efficient coding of stereo pairs. In [73], overlapped block disparity compensation with adaptive search windows is presented. In [34], Moellenhoff *et al.* analyzed the properties of disparity compensated residual images and proposed transform domain coding methods for improved coding of residual images. Other efforts on stereo coding have focused on optimal rate-distortion based algorithms for block dependent quantization [72] and motion/disparity field estimation [64].

Our goal is to develop a compression algorithm, which performs the disparity dependent segmentation of stereo images in an R-D framework. We employ a generalized quadtree decomposition based DE/DC. This scheme further performs the joint coding of the disparity information and the residual image obtained by the disparity compensation in an R-D optimal sense. In the next section, we describe the proposed disparity dependent segmentation based coding scheme for stereo images. We then present some simulation results in Section 6.3.2.

6.3.1 Disparity Dependent Segmentation Based Stereo Image Coding Algorithm

The generic structure of a stereo image coding scheme is as follows: First encode one of the images of the stereo pair as reference, then estimate the disparity information between blocks in the target and encoded reference images and code the disparity information and the residual image obtained by subtracting the disparity compensated target image from the original image. The disparity compensated/predicted target image is obtained by using the encoded reference image and the disparity map. Thus, our scheme employs the closed loop methodology for obtaining the predicted target image.

The main novelty of our stereo image coding scheme is that it performs the disparity dependent quadtree segmentation in an R-D framework.[2] This disparity dependent segmentation based stereo image coding scheme can be described as follows:

[2]Mean squared error (MSE) is used as a distortion measure.

Algorithm 6.2 ***Disparity Dependent Segmentation Based Stereo Image Coding Algorithm***

Step 1: Disparity estimation and compensation

1. Segment the target image using the quadtree decomposition up to a tree depth J.
2. Compute the disparity information for each node by using an intensity based block matching approach.
3. Predict the node using the associated disparity information and encoded reference image.
4. Compute the residual image block by subtracting the predicted target image block from the original image block.
5. Generate the R-D curve for the residual block associated with each node using a standard coding scheme like JPEG2000 [61].

Step 2: The Lagrangian cost $(L(\lambda) = D + \lambda R)$ **based pruning**

6. For the given operating slope $-\lambda$, the 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. This parent-children pruning 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 [45]. This scheme provides a pruned tree. However, this independent pruning scheme fails to join neighboring blocks with similar disparity information, if they have different parents. Thus, this coding scheme fails to exploit the complete dependency among neighbors. For correcting this suboptimality, we introduce the following neighbor joining step.

Step 3: Neighbor joining

7. Given the pruned tree obtained from Step 2, the neighbor joint coding is performed on the leaves of the pruned tree in the R-D optimal manner. That means, the two neighbors 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. This scheme essentially ensures that the neighbors with similar disparity information are jointly coded for further improving the R-D performance. Thus, this leads to the disparity dependent segmentation of the target image.
8. Summing up the allocated rates to all the joint blocks along with the costs of the segmentation map and disparity map will provide the overall bit-rate $R^*(\lambda)$. Similarly, summing up the associated distortions of all the joint blocks will give the net distortion $D^*(\lambda)$. Since we have employed the R-D optimization, $D^*(\lambda)$ represents the minimum distortion for the bit-rate constraint $R^*(\lambda)$.

Step 4: Search for the desired R-D operating slope

9. Similar to Algorithm 2.1, the value for λ is determined iteratively until the bit-rate constraint R_0 is met as closely as possible.

Clearly, the above described algorithm provides the R-D optimal representa-

tion of the target image for a given bit budget and encoded reference image. Due to neighbor joining (Step 3), it is obvious that the proposed scheme results into a non regular partitioning of the target image (for example, see Figure 6.17). It is important to note that the high quality encoded reference image tends to allow for more efficient encoding of the residual image due to the good quality of the disparity predicted target image.

Similar to the encoder, the structure of the decoder is as follows:
1. Reconstruct the reference and residual images.
2. Generate the predicted target image using the disparity information and reconstructed reference image.
3. By summing the predicted target image and residual image, we obtain the desired target image.

6.3.2 Simulation Results: Stereo Image Compression

Simulations are performed on the Arch stereo image pair shown in Figure 6.14.[3] This stereo pair has the disparity predominantly along the horizontal direction, which is usually the case in stereo image coding scenario due to the parallel axis geometry. Therefore, for DE/DC, we utilize a search window of size 64 in the horizontal direction whereas search in the vertical direction is confined only to ± 2 pixels. Figure 6.15 displays the superimposed edge maps for the Arch stereo pair. This figure clearly shows that the objects in one image are shifted horizontally in the other image.

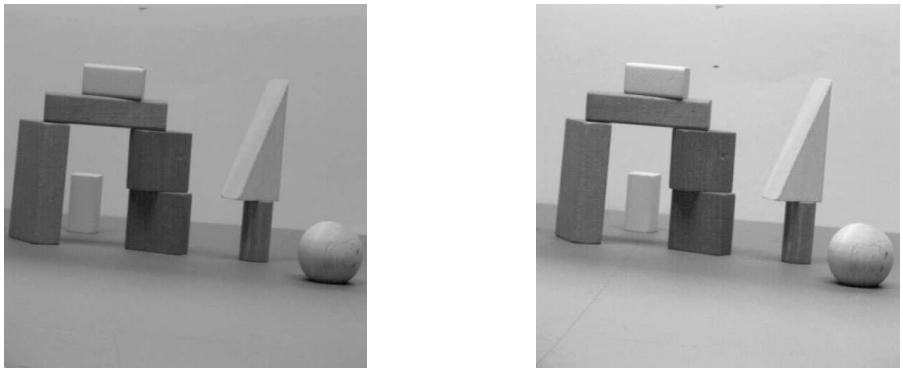
Figure 6.16 shows the disparity field obtained by the quadtree based variable block size decomposition of the target image. As expected, the disparity map is smooth everywhere except along the object boundaries and in the occluded regions.

A reconstructed target image along with the disparity dependent segmentation map is presented in Figure 6.17. It is evident from Figure 6.17 that the algorithm correctly performs finer segmentation along the edges of objects to capture the depth discontinuity precisely, whereas smooth regions with uniform disparity are represented by larger blocks. This results into smaller disparity information overhead bits for smooth regions. This figure also demonstrates that the proposed coding scheme is capable of achieving high compression ratios without sacrificing the reconstructed image quality which we are measuring in terms of PSNR. These simulations also indicate that our R-D optimized algorithm automatically takes care of occluded regions either by performing finer segmentation in those regions or by allocating more bits to corresponding residual blocks.

Figure 6.18 compares the R-D performance of our disparity dependent segmentation based coding scheme with the independent JPEG2000 coder and fixed block size (16×16) based disparity compensated JPEG2000 stereo image

[3] These stereo images are obtained from http://vasc.ri.cmu.edu//idb/html/stereo/arch/index.html.

coder. This R-D performance comparison clearly shows the superiority of the proposed algorithm over the other two algorithms.



(a) Reference image. (b) Target image.

Figure 6.14: Original Arch stereo pair.

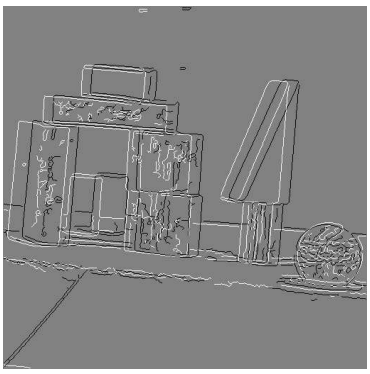


Figure 6.15: Superimposed edge-maps for the Arch stereo pair.

6.4 Discussion

In this chapter, we have addressed two interesting problems, namely denoising and stereo image compression, using tree structured segmentation. In Section 6.2, we have presented a tree based denoising scheme, which can efficiently extract geometrical structures from noisy images. Simulation results shown in Section 6.2.1 clearly demonstrate that our scheme is competitive with the current state of the art wavelet based schemes. Moreover, it is also evident from

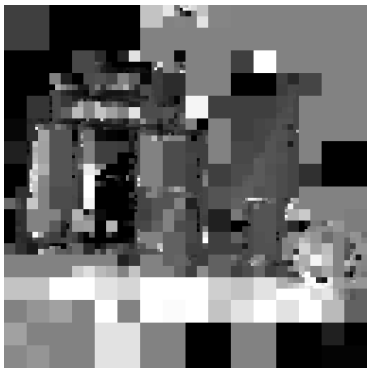


Figure 6.16: Quadtree based disparity map.

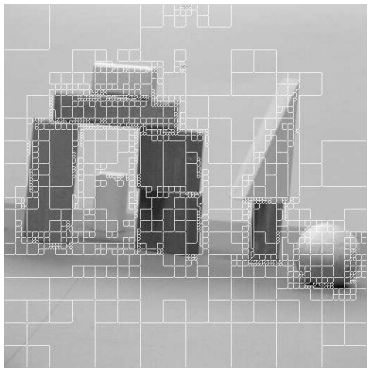


Figure 6.17: Reconstructed target image along with the segmentation map obtained by the disparity dependent segmentation based coding scheme (PSNR=41.4 dB, Bit-rate=0.116 bpp).

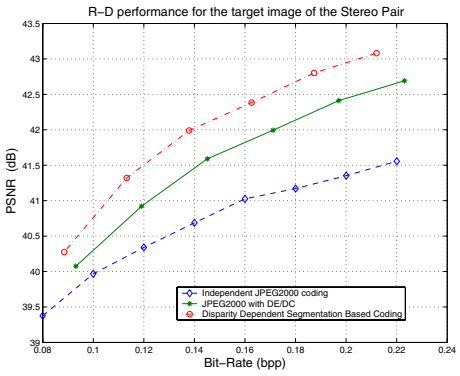


Figure 6.18: R-D performance comparison of different coding schemes for the target image of the Arch stereo pair shown in Figure 6.14.

Figures 6.8 and 6.11 that the proposed scheme captures geometrical features, like edges, of images more precisely compared to wavelet based schemes. Our on-going work aims to extend the present algorithm such that it can also distinguish the texture present in images from random noise for further improving the visual quality of denoised images.

In Section 6.3.1, we have developed an R-D optimized disparity dependent segmentation based stereo image coding scheme. This new scheme performs the joint coding of disparity information and the residual image to achieve the improved R-D performance. Experimental results presented in Figure 6.18 also show that the proposed scheme outperforms the disparity compensated JPEG2000 stereo coder by about 0.5 dB. Since our scheme employs a quadtree based decomposition, this scheme is computationally efficient as well. Our on-going research effort is to extend this stereo image coding scheme to multi-view image and stereo video coding scenarios. We would also like to investigate feature/object based DE/DC scheme in the framework of the proposed algorithm.

Chapter 7

Conclusions

7.1 Summary

In this thesis, we have investigated the problem of coding of piecewise smooth signals/images using tree based segmentation algorithms. The motivating force behind the thesis is to answer the question whether we can design computationally efficient tree based compression algorithms which achieve the optimal R-D behavior for certain simple classes of signals/images. Another important goal of this work has been to understand if algorithms that perform correctly in an R-D sense for some restricted classes of functions can have an impact on the approximation and compression of natural images. The main contributions of the thesis can be summarized as follows.

R-D Optimized Binary Tree Algorithms for 1-D Signals

In Chapters 2 and 3, we studied the 1-D case in detail. We have presented the prune and the prune-join binary tree algorithms in Chapter 2. We have also analyzed the R-D performance of these algorithms for piecewise polynomial signals. This R-D analysis shows that the prune tree algorithm fails to achieve the optimal R-D behavior whereas the prune-join tree algorithm achieves an exponentially decaying R-D behavior $\left(D(R) \sim c_4 2^{-c_5 R}\right)$ due to the neighbor joint coding strategy. Most importantly, these schemes are computationally efficient as well. In Chapter 3, we performed the R-D analysis for 1-D piecewise smooth functions. In particular, we have shown that both the prune and the prune-join schemes achieve the polynomially decaying asymptotic R-D behavior $\left(D(R) \sim d_0 \left(\frac{\log R}{R}\right)^{2\alpha}\right)$ for piecewise smooth signals. Simulation results also show that the R-D behaviors of the two coding schemes are consistent with the theory.

R-D Optimized Quadtree Algorithms for 2-D Signals

In Chapter 4, we have extended our binary tree based coding algorithm to the 2-D case in the form of quadtree based coding algorithm. Similar to 1-D, we have also proved that the prune-join quadtree coding algorithm achieves an exponentially decaying asymptotic R-D behavior for the piecewise polynomial image model. The computational complexity of the quadtree algorithm is also shown to be $O(N \log N)$. In Chapter 5, we then analyzed the R-D behavior for more general piecewise smooth images. Specifically, we have proved that both the prune and the prune-join schemes achieve the polynomially decaying asymptotic R-D behavior for the 2-D smooth image model and the Horizon image model. Experimental results have also confirmed that the algorithm achieves optimal performance if the input image fits the model exactly. Moreover, simulations showed that our quadtree algorithm consistently outperformed wavelet based coder (JPEG2000) also in case of compression of real images, like cameraman, at low rates.

Geometrical Image Denoising and Stereo Image Compression

In Chapter 6, two interesting image processing problems, namely denoising and stereo image compression, are considered in the framework of tree structured segmentation. For the denoising problem, we have outlined a tree based algorithm which performs denoising by compressing the noisy signal or image. We have performed numerical simulations which show that our compression based denoising scheme achieves improved visual quality by capturing geometrical features, like edges, of images more precisely compared to wavelet based schemes. We then addressed the problem of stereo image compression and proposed a novel rate-distortion optimized disparity based coding scheme for stereo images. We have also presented simulation results for the proposed scheme and compared these results with the performance of a fixed block size DE/DC based JPEG2000 stereo image coder.

7.2 Future Research

The development of new coding methods for images that can capture geometrical features with low computational cost is one of the most active research areas in the image processing community. The tree structured segmentation based coding algorithms presented in this thesis are very promising in extracting the geometrical structure present in images. Since geometrical features, like edges, represent one of the most important perceptual information in an image, the proposed algorithms can play an important role in applications, such as computer vision and graphics, requiring efficient representation of geometrical information present in images. In the following, we provide an overview of ongoing and future research directions.

Search for Globally Optimal Solution

Consider a discontinuous function $f(t)$ which is generated by multiplying a polynomial $p(t)$ with an on-off square pulse train. This function is either zero or characterized by only one polynomial $p(t)$. An oracle based scheme observes this fact and codes the signal accordingly to obtain the globally optimal solution. However, our tree based scheme, which is presently able to exploit the neighbor similarity only, cannot observe this phenomenon hidden in the signal. So clearly the tree scheme cannot perform as well as the oracle method. However, since the generated signal has only finite degrees of freedom, the tree scheme still achieves an exponentially decaying R-D behavior. But the rate of decay is smaller in comparison to that of the oracle method. An interesting direction of work would be to incorporate the concept of global similarity in the framework of the tree algorithm so that the tree scheme can match the performance of the oracle method for the above described synthetic signals as well.

Texture Modeling

The main focus of this thesis remains the efficient representation of smooth surfaces and smooth edges separating the smooth regions without much attention to the proper modeling/coding of texture present in natural images. Since natural images can be more accurately modeled as a sum of 2-D piecewise smooth function and texture [24], we need to incorporate efficient texture modeling to achieve improved performance for natural images at high rates. It is well known that wavelets can model texture better than piecewise polynomials. Thus, the quadtree scheme should model tree nodes with texture using wavelets and tree nodes with geometrical information using a geometrical tile composed of two 2-D polynomials separated by a linear edge. In this way, this new scheme will enjoy the best of both wavelets modeling and geometrical modeling and hopefully achieve a better R-D performance.

Multi-view Image and Stereo Video Coding

For the problem of stereo image compression, we have seen that the proposed disparity dependent segmentation based scheme outperforms JPEG2000 based stereo coder. Therefore, it is of interest to extend this stereo image coding scheme to multi-view image and stereo video coding scenarios. In particular, using feature/object based DE/DC and ME/MC schemes in the framework of the proposed algorithm may provide improved performance.

Distributed Stereo Image Coding

Another interesting problem setup is the distributed coding scenario for stereo images. For instance, consider an electronic surveillance system where the two

cameras are viewing the same scene and sending the images to the central security unit for further processing. In this setup, communication between the two encoders is not allowed due to some constraints such as power constraint. However, both encoders have the information about the correlation structure of the observed entities. In this scenario, an open issue is to understand whether we can design distributed stereo coding schemes which can perform as well as the traditional stereo coding schemes.

Bibliography

[1] H. Aydinoglu and H. Hayes, "Stereo image coding: A projection approach," *IEEE Trans. Image Proc.*, vol. 7, pp. 506-516, Apr. 1998.

[2] D. H. Ballard and C. M. Brown, *Computer Vision*, Prentice-Hall, Englewood Cliffs NJ, 1982.

[3] L. Balmelli, M. Vetterli and T. Liebling, "Mesh optimization using global error with application to geometry simplification," Graphical Models 64, 230-257 (2002), Special Issue on Processing of Large Polygonal Meshes, Academic Press.

[4] N. V. Boulgouris and M. G. Strintzis, "A family of wavelet-based stereo image coders," *IEEE Trans. Circuits Syst. Video Tech.*, vol. 12, no. 10, pp. 898-903, Oct. 2002.

[5] E. J. Candes and D. L. Donoho, "Curvelets- a surprisingly effective nonadaptive 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.

[6] S. G. Chang, B. Yu, M. Vetterli, "Spatially adaptive image wavelet thresholding with context modeling for image denoising," *IEEE Trans. on Image Proc.* 9(9):1522-1531, Sep. 2000

[7] S. G. Chang, B. Yu, M. Vetterli, "Wavelet thresholding for multiple noisy image copies," *IEEE Trans. Image Proc.*, 9(9):1631-1635, Sep. 2000

[8] 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.

[9] 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.

[10] 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.

[11] R.R. Coifman and D.L. Donoho, "Translation invariant denoising," Tech. Rep., Department of Statistics, Stanford University, May 1995.

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

[13] S. Dekel, D. Leviatan and M. Sharir, "On bivariate smoothness spaces associated with nonlinear approximation," Constructive Approximation (to appear).

[14] 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.*, Thessaloniki, Greece, Oct. 2001.

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

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

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

[18] D.L. Donoho and I.M. Johnstone, "Ideal Spatial adaptation via wavele t shrinkage," Biometrika, 81:425-455, Dec. 1994.

[19] D.L. Donoho, M. Vetterli, R.A. DeVore, and I. Daubechies, "Data compression and harmonic analysis," *IEEE Trans. Inform. Th.,* vol. 44, no. 6, pp. 2435-2476, Oct. 1998.

[20] P. L. Dragotti, *Wavelet Footprints and Frames for Signal Processing and Communications*, PhD Thesis, Department of Communication Systems, Swiss Federal Institute of Technology Lausanne, April 2002.

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

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

[23] 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.

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

[25] J. Jiang and E. A. Edirisinghe, "A hybrid scheme for low bit-rate coding of stereo images," *IEEE Trans. Image Proc.*, vol. 11, no. 2, pp. 123-134, Feb. 2002.

[26] A. P. Korostelev and A. B. Tsybakov, *Minimax theory of image reconstruction*, Springer-Verlag, New York, 1993.

[27] F. Labonte, C. T. Le Dinh, J. Faubert, and P. Cohen, "Spatiotemporal spectral coding of stereo image sequences," *IEEE Trans. Circuits, Syst., Video Tech.*, vol. 9, pp. 144-155, Feb. 1999.

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

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

[30] T. Lookabaugh and R.M. Gray, "High resolution quantization theory and the vector quantizer advantage," *IEEE Trans. Inform. Th.,* vol. 35, pp. 1020-1033, Sep. 1989.

[31] M. E. Lukacs, "Predictive coding of multi-viewpoint image sets," in *Proc. ICASSP*, Oct. 1986, pp. 521-524.

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

[33] S. Mallat and W.L. Hwang, "Singularity detection and processing with wavelets," *IEEE Trans. on Inform. Th.*, 38:617-643, March 1992.

[34] M. S. Moellenhoff and M.W. Maier, "Transform coding of stereo image residuals," *IEEE Trans. Image Proc.*, vol. 7, pp. 804-812, June 1998.

[35] B. K. Natarajan, "Filtering Random Noise from Deterministic Signals via Data Compression," *IEEE Trans. Signal Proc.*, vol. 43(11), pp. 2595-2605, Nov. 1995.

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

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

[38] M. G. Perkins, "Data compression of stereo pairs," *IEEE Trans. Comm.*, vol. 40, pp. 684-696, Apr. 1992.

[39] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli, "Image denoising using gaussian scale mixtures in the wavelet domain," *IEEE Trans. Image Proc.*, vol. 12(11), pp. 1338 - 1351, Nov. 2003.

[40] 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.

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

[42] A. Puri, R. V. Kollarits, and B. G. Haskell, "Basics of stereoscopic video, new compression results with MPEG-2 and a proposal for MPEG-4," *Jour. Signal Proc. Image Comm.*, vol. 10, pp. 201-234, 1997.

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

[44] 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.

[45] 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.

[46] E.A.Riskin, "Optimal bit allocation via the generalized BFOS algorithm," *IEEE Trans. Inform. Th.,* vol. 37, pp. 299-315, Mar. 1991.

[47] J. K. Romberg, H. Choi, and R. G. Baraniuk, "Bayesian tree-structured image modeling using wavelet domain hidden markov models," *IEEE Trans. Image Proc.*, vol. 10, pp. 1056-1068, no. 7, July 2001.

[48] A. Segall, " Bit allocation and encoding for vector sources," *IEEE Trans. Inform. Th.,* vol. IT-22, pp. 162-169, Mar. 1976.

[49] S. Sethuramam, M. W. Siegel, and A. G. Jordan, "A multi-resolution framework for stereoscopic image sequence compression," in *Proc. ICIP-95*, vol. 2, pp. 361-365.

[50] Y. Shoham and A. Gersho, " Efficient bit allocation for an arbitrary set of quantizers," *IEEE Trans. Acoust. Speech Signal Proc.*, vol. 36, pp. 1445-1453, Sep. 1988.

[51] R. Shukla, M.N. Do, P. L. Dragotti and M. Vetterli, "Rate-distortion optimal tree based coding algorithms for piecewise polynomials," in *Proc. IEEE Int. Symp. on Information Theory*, Switzerland, July 2002.

[52] 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.

[53] R. Shukla, P. L. Dragotti, M.N. Do and M. Vetterli, "Rate-distortion optimized tree based coding algorithms," Proc. of IEEE Int. Workshop on Information Th., India, October 2002.

[54] R. Shukla, P. L. Dragotti, M.N. Do and M. Vetterli, "Rate-distortion optimized tree structured compression algorithms for piecewise smooth images," accepted to *IEEE Trans. Image Proc.*, Mar. 2004.

[55] Rahul Shukla, Hayder Radha and M.Vetterli, "Disparity Dependent Segmentation Based Stereo Image Coding," in *Proc. IEEE Int. Conf. on Image Proc.*, Barcelona, Sep. 2003.

[56] Rahul Shukla and M.Vetterli, "Geometrical Image Denoising Using Quadtree Segmentation," submitted to *IEEE Int. Conf. on Image Proc.*, Singapore, Oct. 2004.

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

[58] M. G. Strintzis and S. Malassiotis, "Object-based coding of stereoscopic and 3D image sequences: A review," *IEEE Signal Proc. Mag.*, vol. 16, pp. 14-29, May 1999.

[59] P. Strobach, "Quadtree structured recursive plane decomposition coding of images," *IEEE Trans. Signal Proc.*, vol. 39, pp. 1380-1397, June 1991.

[60] 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.

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

[62] H. Triebel, *Interpolation Theory, Function Spaces, Differential Operators*, North-Holland, Amsterdam, 1978.

[63] D. Tzovaras, N. Grammalidis, and M. G. Strintzis, "Object-based coding of stereo image sequences using joint 3D motion/disparity compensation," *IEEE Trans. Circuits Syst. Video Tech.*, vol. 7, pp. 312-327, Apr. 1997.

[64] D. Tzovaras and M. G. Strintzis, "Motion and disparity field estimation using rate-distortion optimization," *IEEE Trans. Circuits Syst. Video Tech.*, vol. 8, pp. 171-180, Apr. 1998.

[65] 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.

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

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

[68] M.Vetterli and J. Kovačević, *Wavelets and Subband Coding*, Prentice Hall, Englewood Cliffs, NJ, 1995.

[69] V. Velisavljevic, B. Beferull-Lozano, M. Vetterli, and P. L. Dragotti, "Discrete Multi-directional Wavelet Bases," *IEEE Int. Conf. on Image Proc.* , Barcelona, Sep. 2003.

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

[71] R. Willett and R. Nowak, "Platelets: a multiscale approach for recovering edges and surfaces in photon-limited medical imaging," IEEE Trans. on Medical Imaging, Vol. 22, no. 3, pp. 332 -350, March 2003.

[72] W. Woo and A. Ortega, "Optimal blockwise dependent quantization for stereo image coding," *IEEE Trans. Circuits Syst. Video Tech.*, vol. 9, pp. 861-867, Sept. 1999.

[73] W. Woo and A. Ortega, "Overlapped block disparity compensation with adaptive windows for stereo image coding," *IEEE Trans. Circuits Syst. Video Tech.*, vol. 10, pp. 194-200, Mar. 2000.

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

[75] C. Q. Zhan, L. J. Karam, "Wavelet-based adaptive image denoising with edge preservation," *IEEE Int. Conf. on Image Proc.*, Barcelona, Sep. 2003.

Curriculum Vitae

Rahul Shukla
Audio-Visual Communications Laboratory
Swiss Federal Institute of Technology Lausanne (EPFL)
1015 Lausanne, Switzerland
Email: `rahul.shukla@epfl.ch`
Web: `http://lcavwww.epfl.ch/~rahul`

Education

2000 - 2004 Ph.D. candidate in Department of Computer and Communication Sciences, Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland.

1999 - 2000 Doctoral School in Communication Systems, EPFL.

1995 - 1999 Bachelor of Technology in Electrical Engineering, Indian Institute of Technology, Kanpur, India.

Professional Experience

2000 - 2004 **Research and Teaching Assistant**, Audio-Visual Communications Laboratory, Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland.

Summer 2002 **Summer Research Scholar** in Wireless and Video Communication Lab, Michigan State University, East Lansing, MI, USA.

Awards and Achievements

- Fellowship from the doctoral school of communication systems, Swiss Federal Institute of Technology Lausanne (1999-2000).

- Secured 1^{st} rank in the Communication Systems doctoral program (1999-2000) at Swiss Federal Institute of Technology, Lausanne.
- Tata Consultancy Services award for the best undergraduate project work in the field of Electrical Engineering in 1999 at Indian Institute of Technology, Kanpur.
- Summer Under Graduate Design Fellowship in 1998 by the "R & D wing" of Indian Institute of Technology Kanpur for designing "Fiber-optic Communication Receiver Module".
- Secured 68th ALL INDIA RANK among the hundred thousand participants in the Joint Entrance Exam of Indian Institute of Technology (J.E.E.95).
- Secured 06th ALL INDIA RANK among the fifty thousand participants in the Combined Entrance Exam of Regional Engineering Colleges (C.E.E.95).
- Received state merit scholarship for the duration of four years.

Personal

Date of birth:	February 07, 1978.
Nationality:	Indian.
Civil-status:	Bachelor.

Publications

Journals and Manuscripts

1. R. Shukla, P.L.Dragotti, M.Do, and M.Vetterli, "Rate Distortion Optimized Tree Structured Compression Algorithms for Piecewise Smooth Images", accepted to the IEEE Trans. on Image Processing, March 2004.
2. R. Shukla, P.L. Dragotti, M. Do and M. Vetterli, "Rate-Distortion Bounds of Tree Based Coding Algorithms for Piecewise Smooth Functions", to be submitted to the IEEE Trans. on Signal Processing.

Conferences

1. R. Shukla and M. Vetterli "Geometrical Image Denoising Using Quadtree Segmentation," submitted to *IEEE Int. Conf. on Image Proc.*, Singapore, Oct. 2004.

2. R. Shukla, H. Radha and M. Vetterli "Disparity Dependent Segmentation Based Stereo Image Coding", in Proc. of *IEEE Int. Conf. on Image Proc.*, Barcelona , September 2003.

3. R. Shukla, P.L.Dragotti, M.Do, and M.Vetterli, "Rate-Distortion Optimized Tree Based Coding Algorithms", in Proc. of *IEEE Int. Workshop on Inform. Th.*, India , October 2002.

4. R. Shukla, P.L.Dragotti, M.Do, and M.Vetterli, "Improved Quadtree Algorithm Based on Joint Coding for Piecewise Smooth Image Compression", in Proc. of *IEEE Int. Conf. on Multimedia*, Switzerland, Aug. 2002.

5. R. Shukla, M.Do, P.L.Dragotti, and M.Vetterli, "Rate-Distortion Optimal Tree Algorithms for Piecewise Polynomials", in Proc. of *IEEE Int. Symposium on Inform. Th.*, Switzerland , July 2002.

6. M.Do, P.L.Dragotti, R.Shukla, and M.Vetterli, "On the Compression of Two-Dimensional Piecewise Smooth Functions", invited paper, in Proc. of *IEEE Int. Conf. on Image Proc.*, Thessaloniki, Greece, October 2001.

7. R. Shukla, M. N. Do and M.Vetterli, "Best Adaptive Tiling in a Rate-Distortion Sense", *MSRI workshop on NonLinear Estimation and Classification*, Berkeley, California, USA, March 2001.