Método Robusto e Simples de Compressão de Imagens Baseado no Algoritmo EZW

por
Andrezza Almeida Gusmão

Dissertação de Mestrado apresentada à
Universidade Estadual de Campinas – Unicamp,
Faculdade de Engenharia Elétrica e de Computação,
como requisito parcial para a obtenção do grau de
Mestre em Engenharia Elétrica

Orientador:
Prof. Dr. Max Henrique Machado Costa

Banca Examinadora:
Prof. Dr. Nelson Delfino D'Ávila Mascarenhas - UFSCar
Prof. Dr. Dalton Soares Arante – FEEC/Unicamp
Prof. Dr. Amauri Lopes – FEEC/Unicamp

Campinas, São Paulo
Março de 2002

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

G972m

Gusmão, Andrezza Almeida
Método robusto e simples de compressão de imagens baseado no algoritmo EZW / Andrezza Almeida Gusmão.--Campinas, SP: [s.n.], 2002.

Orientador: Max Henrique Machado Costa.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Wavelets (Matemática). 2. Compressão de imagens. 3. Processamento de imagens. I. Costa, Max Henrique Machado. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Não siga aonde leva a trilha.
Em vez disso, vá aonde não
há trilhas e deixe seu rastro.

Resumo

A proliferação de serviços de comunicação sem fio e baseados na Internet, juntamente com a demanda por produtos de multimídia, tem estimulado o interesse na transmissão de imagens e de vídeo em canais ruidosos de comunicação cuja capacidade pode variar com o tempo. Neste trabalho propõe-se um algoritmo de compressão de imagens implementado especialmente para este tipo de aplicação, onde há uma maior necessidade de proteção contra erros durante a transmissão. O algoritmo proposto apresenta uma certa robustez a erro de bits de transmissão. Ele é uma versão modificada do algoritmo REZW (*Robust Embedded Zerotree Wavelet*) proposto por Creusere [33], em que a codificação aritmética é substituída por codificação de Huffman, com a vantagem de redução de complexidade. Sua robustez é alcançada pela divisão dos coeficientes *wavelets* em diferentes seqüências que são quantizadas e codificadas independentemente antes da transmissão. Deste modo, um erro de bit em uma das seqüências não afeta as outras seqüências, permitindo que mais informação correta chegue ao decodificador. O trabalho apresenta resultados de simulações e comparações de desempenho.

Abstract

The growth of wireless communication services and internet-based services, along with consumer demand for multimedia products, has spurred interest in the transmission of image and video signals over noisy communication channels with time varying capacities. This work proposes an image compression algorithm specially implemented for this type of application, where there is an increased need for protection against errors. The algorithm proposed is a modification of Creusere REZW (Robust Embeeded Zerotree Wavelet) algorithm [33], where arithmetic coding is substituted by Huffman coding, with the advantage of a reduction in complexity. This algorithm achieves robustness against transmission errors by partitioning the transform coefficients into groups and independently coding and transmitting each group using an embedded encoder. In this way, a bit error in one of the groups does not affect others groups, allowing more uncorrupt information to reach the decoder. The work presents simulation results and performance comparisons.

Agradecimentos

Agradeço ao meu orientador Max Henrique Costa, cujo conhecimento, competência e dedicação foram essenciais para o êxito deste trabalho. Agradeço especialmente o apoio, seriedade e amizade constantes que me dedicou durante estes anos.

Meus agradecimentos ao Professor Amauri Lopes pela sua ajuda nas técnicas de wavelets que foram bastante úteis na iniciação deste trabalho. Ao Professor Dalton Soares Arantes pelo seu apoio e amizade.

Aos funcionários da Unicamp pelo suporte técnico e administrativo. Em especial à Lúcia, secretária do Decom, pela sua amizade e prestatividade.

Ao CNPq pelo suporte financeiro às várias fases deste trabalho.

Aos amigos que fiz em Campinas, em especial, Lais, Juliana, Flávia, Rinaldo, Helder, Romis, Cynthia. Suas presenças na minha vida tornaram mais fácil suportar a saudade da família. Obrigada a todos vocês que são meus verdadeiros amigos.

Ao Romis que muito me ajudou na implementação dos algoritmos deste trabalho. Agradeço pelo tempo que me foi dedicado tão atenciosamente.

Aos meus familiares que muito me apoiaram na realização deste projeto, em especial Tio Waltinho, Tia Márcia, Tia Bete, Tio Renato, Tia Maria Inês, Tio Moacyr e a minha Vó.

Agradeço de uma forma muito especial aos meus pais que mesmo longe sempre estiveram presentes na minha vida, me apoiando e me aconselhando nas horas que mais precisava. Agradeço por sempre terem demonstrado o amor que sentem por mim. Ao meu irmão por sempre estar disposto a me ajudar nos momentos que precisei.

Agradeço ao Aldo, Dora e Gi, por terem se tornado minha família aqui em Campinas. Por todo carinho e atenção que sempre me dedicaram.

Ao meu companheiro Dani, pelo apoio, carinho e compreensão que me dedicou durante todos estes anos, sem os quais eu teria sucumbido nos momentos de insegurança. A tarefa de terminar este trabalho teria sido infinitamente mais árdua sem a sua presença. Ao meu companheiro, meu eterno amor.

A todos vocês meu muito obrigada.

Andrezza.

ÍNDICE

Lista de Figuras e Tabelas

Capítulo 1

Introdução

A informação audiovisual tem sido transportada principalmente através de redes específicas, mas a tendência atual é em direção à generalização do transporte da informação audiovisual juntamente com a informação proveniente de sinais de voz, de áudio e de dados em uma rede de serviços integrados de telecomunicações, tanto no caso de redes sem fio quanto no caso de redes com fibra óptica. Esta tendência se deve à demanda de taxas cada vez maiores para atender os novos serviços de telecomunicações. Para que estas informações sejam transportadas requer-se uma maior largura de banda para sua transmissão ou armazenamento e uma maior robustez a erros (maior segurança) durante a transmissão.

Neste trabalho investiga-se um algoritmo de compressão de imagens implementado especialmente para ser usado em canais de comunicação sem fio, onde há uma maior necessidade de proteção contra erros durante a transmissão. O algoritmo proposto é uma versão modificada do algoritmo REZW (*Robust Embedded Zerotree Wavelet*) estudado por Creusere [33]. O algoritmo REZW, por sua vez, é baseado no algoritmo EZW (*Embedded Zerotree Wavelet*) investigado por Shapiro [32], e apresenta uma maior robustez a erros de bits que o algoritmo EZW. Esta robustez é alcançada pela divisão dos coeficientes *wavelets* em diferentes seqüências (vetores) que são quantizadas e codificadas independentemente. Estas seqüências de bits são então intercaladas (*interleaved*) como apropriado (por exemplo, em bits, bytes, pacotes, etc.), antes da transmissão, de forma que a natureza embutida do vetor de bits seja mantida. A natureza embutida do vetor de bits produzida pelo codificador provê um certo grau de proteção contra erros durante a transmissão. Toda a informação que surge antes do primeiro erro de bit ocorrer pode, tipicamente, ser usada para reconstruir uma versão aproximada da imagem. No entanto, tudo o que ocorre depois do erro é perdido. Este é um contraste direto com algoritmos não embutidos de compressão onde um único erro pode irreparavelmente comprometer a qualidade da imagem.

A característica mais importante dos codificadores embutidos é a propriedade de transmissão progressiva. Transmissão progressiva refere-se à transmissão da informação na

ordem decrescente de importância. Em outras palavras, os coeficientes com magnitudes maiores são transmitidos antes dos coeficientes de magnitudes menores. Tal esquema de transmissão faz com que os vetores de bits sejam embutidos (*embedded*), i.e., um único código pode ser usado para decodificar a imagem em taxas menores ou iguais à taxa codificada, obtendo a melhor reconstrução possível da imagem. A natureza embutida do vetor de bits, faz com que seja possível ao codificador interromper a codificação quando uma desejada taxa de compressão for atingida ou alguma outra condição for satisfeita. Esta característica é importante em termos de largura de banda, espaço de armazenamento, complexidade computacional, tempo e custo.

Codificadores com estas características têm se tornado cada vez mais populares em compressão de imagens por apresentar um excelente desempenho e baixa complexidade computacional, além das características de transmissão progressiva e codificação embutida. Estes algoritmos obtêm resultados que são competitivos com praticamente todos os algoritmos de compressão de imagens conhecidos. Seu bom desempenho é alcançado sem requerer conhecimento a priori da imagem ou tabelas de codificação previamente armazenadas como na quantização vetorial, além de não requerer nenhum treinamento dos codificadores e decodificadores, como no padrão JPEG (para otimizar as tabelas de quantização).

O algoritmo apresentado neste trabalho, denominado algoritmo modificado, difere do algoritmo REZW, estudado por Creusere [33], na escolha do codificador de entropia. Enquanto o algoritmo REZW utiliza um codificador aritmético, o algoritmo modificado usa um codificador de Huffman. Os desempenhos apresentados por este algoritmo são ligeiramente inferiores aos apresentados pelo algoritmo REZW, devido ao uso deste codificador de entropia, mas com a vantagem de apresentar maior simplicidade computacional. Além disso, o codificador aritmético é protegido por patentes enquanto que o codificador de Huffman é de domínio público.

O Capítulo 2 apresenta uma introdução à teoria de compressão utilizando *wavelets* onde se enfatizam os aspectos da transformada *wavelet* julgados mais importantes para a compressão de imagens. No Capítulo 3 é apresentado o algoritmo de compressão de imagens EZW (*Embedded Zerotree Wavelet*) [32]. Este algoritmo explora a natureza de multiresolução da decomposição *wavelet* criando uma forma totalmente nova de codificar imagens. É citado também o algoritmo REZW (*Robust Embedded Zerotree Wavelet*) [33], que é a base deste trabalho. Este algoritmo é uma extensão do algoritmo EZW. O Capítulo 4 apresenta o algoritmo

proposto, as curvas e as imagens resultantes das simulações realizadas com o algoritmo proposto. Apresentam-se também os resultados dos algoritmos REZW e EZW, retirados dos trabalhos de Creusere [33] e de Shapiro [32], respectivamente. Estes resultados são apresentados como meios de comparação entre estes algoritmos e o algoritmo modificado. Por fim, o Capítulo 5 apresenta comentários, conclusões e sugestões para trabalhos futuros.

Capítulo 2

Compressão de Imagens Utilizando *Wavelets*

Tem sido crescente o interesse em pesquisar métodos de codificação de imagens que apresentem altas taxas de compressão. O principal objetivo da compressão de imagem é representá-la com a menor quantidade de bits possível, preservando a qualidade da imagem exigida para uma dada aplicação.

Um dos métodos mais utilizados na compressão de imagens é a codificação por transformada. No domínio da transformada, verifica-se uma redução na correlação existente entre os elementos de uma determinada imagem (pixels), e procura-se concentrar a maior quantidade de energia possível no menor número de coeficientes. Como estes coeficientes tendem a apresentar correlação reduzida, eles podem ser quantizados separadamente.

Entre as transformadas utilizadas para compressão de imagens destacam-se as Transformadas *Wavelet* (T.W.) [1], que apresentam uma grande capacidade de concentrar energia. Esta classe de transformadas apresenta uma estreita relação com o sistema visual humano, o que permite obter uma alta taxa de compressão com menor degradação percebida. Uma das propriedades mais atraentes das transformadas *wavelet* é a sua flexibilidade com relação à duração e à posição das funções de base da representação (*wavelets*), o que permite que sinais com conteúdos de freqüência variáveis, como imagens, possam ser bem representados com um número reduzido de funções. Além disso, a utilização da transformada *wavelet* diminui sensivelmente o efeito de bloco (*blocking effect*), que é um dos principais problemas do padrão JPEG (*Joint Picture Experts Group*), baseado na transformada discreta do co-seno (DCT). *Wavelets* longas podem ser utilizadas para representar, por exemplo, as regiões planas de uma imagem (baixa freqüência), enquanto as *wavelets* mais estreitas são utilizadas para representar regiões de textura (alta freqüência) [2], [3].

Neste capítulo será apresentado cada um dos estágios de um esquema básico de compressão baseado na transformada *wavelet*.

2.1 Sistema Básico de Compressão Baseado na Transformada *Wavelet*

O diagrama de blocos de um sistema básico de compressão de imagens baseado na transformada *wavelet* é apresentado na Fig. 2.1-a. Este sistema pode ser dividido em três estágios: a transformada *wavelet*, a quantização e a codificação de entropia [4]. Cada um destes estágios é de grande importância no desempenho da compressão e o seu projeto deve levar em conta as características da imagem, as taxas de compressão desejadas e as limitações inerentes ao processo de implementação. O processo inverso é realizado no receptor para a reconstrução da imagem conforme apresentado na Fig. 2.1-b.

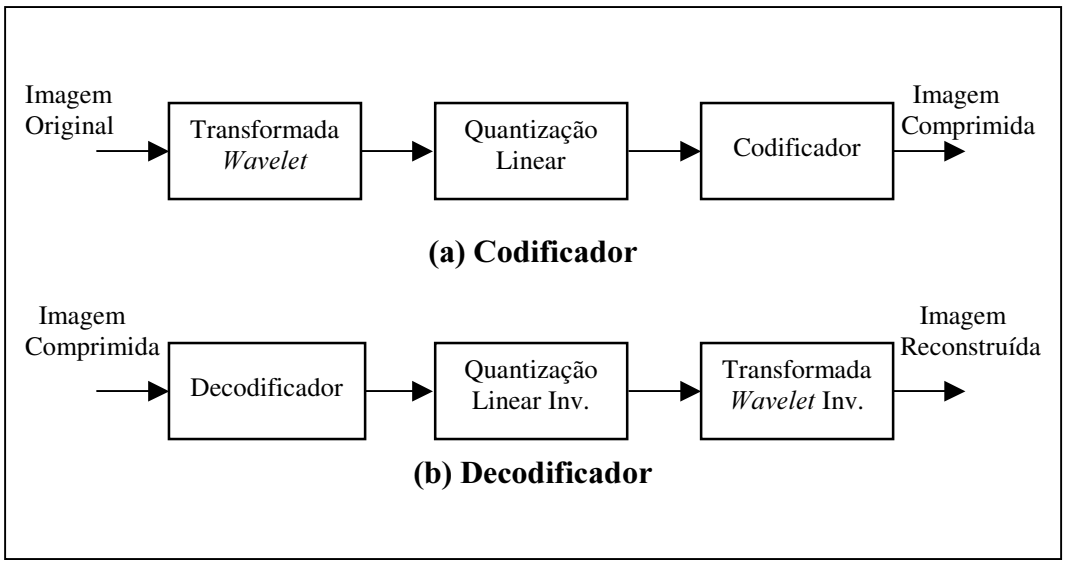


Fig. 2.1: Diagrama de blocos de um esquema básico de compressão baseado na Transformada *Wavelet*.

No primeiro estágio calcula-se a transformada *wavelet* bidimensional da imagem onde ela pode ser decomposta em uma, duas ou mais camadas. Além disso, existe uma grande variedade de famílias de *wavelets* que podem ser utilizadas. Cada uma delas apresenta suas características especificas como, por exemplo, suavidade, ortogonalidade, etc.

No segundo estágio, os coeficientes da transformada são quantizados. A quantização tem como objetivo limitar o número de possíveis valores para a representação dos coeficientes gerados pelas transformadas a um conjunto finito. A operação de quantização torna o processo de compressão irreversível, i.e., a imagem reconstruída é numericamente diferente da imagem original devido à introdução do erro de quantização.

Após o processo de quantização, pode-se ainda explorar a redundância estatística presente nos dados, usando as técnicas de codificação sem perda ou codificação de entropia [5]. Entre os códigos de entropia existentes, os mais utilizados são o código de Huffman [6] e o código Aritmético [7], [8]. Neste trabalho será utilizado o código de Huffman por se tratar de um código simples e eficiente.

2.2 A Transformada *Wavelet*

A teoria da decomposição de sinais em *wavelets* é relativamente recente e se desenvolveu especialmente nos últimos 15 anos através das contribuições de pesquisadores das mais diversas áreas como matemática, física, estatística, computação gráfica e engenharia. Sob o ponto de vista de processamento de sinais, pode-se pensar na transformada *wavelet* como uma alternativa para as técnicas tradicionais de análise de sinais, como a análise de *Fourier* e a *Short Time Fourier Transform* (STFT). Bons resultados têm sido obtidos com a aplicação de *wavelet* em diferentes áreas como compressão de imagens, processamento de sinais, processamento de voz, visão computacional e outros [4], [9], [10].

O objetivo deste capítulo é dar uma introdução sucinta à teoria de transformada *wavelet*. Maiores detalhes sobre este assunto podem ser encontrados em [11], [12], [13] e [14].

2.2.1 A Transformada *Wavelet* Contínua (TWC)

A transformada *wavelet* mapeia uma função unidimensional em uma função bidimensional de variáveis *a* e *b*. O parâmetro *a*, de escala, é responsável pela compressão ou expansão da função. O parâmetro *b* é a translação da função *wavelet* ao longo da escala de tempo. Considerando-se que um sinal real *s(t)* pertence à classe de funções reais mensuráveis, denotada como $s(t) \in \mathbf{L}^2(\Re)$, de forma que

$$\int_{-\infty}^{+\infty} s^2(t)dt < \infty, \tag{2.1}$$

a Transformada *Wavelet* Contínua de *s(t)* é dada por [15]

$$TWC(a,b) = \Psi(a,b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{+\infty} s(t)\psi\left(\frac{t-b}{a}\right)dt, \tag{2.2}$$

onde $\psi(t)$ é a *wavelet* mãe ou *wavelet* base e $\psi((t-b)/a)/\sqrt{|a|}$ são as variações da *wavelet* base, também chamadas *wavelets* filhas. A *wavelet* mãe pode ser real ou complexa. Quando $\psi(t)$ é complexa, usa-se o conjugado de $\psi(t)$ na Eq. 2.2. Para algumas aplicações, pode ser vantajoso o uso de *wavelets* complexas. Neste trabalho utilizam-se exclusivamente as *wavelets* reais.

Em qualquer aplicação que utilize transformada, é importante que haja inversibilidade. Se $\psi(t)$ é tal que permite a inversibilidade da transformada, então a Transformada *Wavelet* Contínua Inversa é dada por

$$s(t) = \frac{1}{c_\psi} \int_{-\infty a>0}^{+\infty}\int^{+\infty} \Psi(a,b)\frac{1}{\sqrt{|a|}}\psi\left(\frac{t-b}{a}\right)\frac{1}{a^2}dadb, \tag{2.3}$$

onde c_ψ é uma constante que depende da *wavelet* usada. O sucesso da reconstrução depende desta constante c_ψ, que é chamada constante de admissibilidade.

A condição de admissibilidade limita a classe de funções que podem ser utilizadas como *wavelets* mãe e é dada por

$$c_\psi = \int_{-\infty}^{+\infty} \frac{\left|\Psi(\xi)\right|^2}{|\xi|} d\xi < \infty, \tag{2.4}$$

onde $\Psi(\xi)$ é a Transformada de *Fourier* de $\psi(t)$, $\Psi(\xi) = \int_{-\infty}^{+\infty} \psi(t).e^{-j2\pi\xi t}dt$. Sabendo que $\psi(t)$ é uma função finita e de suporte compacto, tem-se

$$\int_{-\infty}^{+\infty} |\psi(t)|dt < \infty, \tag{2.5}$$

de tal forma que $\Psi(\xi)$ é contínua em $\Re$ [11].

Dada a Eq. 2.4 tem-se que $\Psi(\xi)$ deve se anular na origem, $\Psi(0) = 0$. Assim,

$$\Psi(0) = \int_{-\infty}^{+\infty} \psi(t)dt = 0. \tag{2.6}$$

A Transformada *Wavelet* Contínua mapeia uma função real de uma variável real $s(t)$, $-\infty < t < \infty$, em uma função real de duas variáveis reais TWC(a,b), $-\infty < a,b < \infty$, $a \neq 0$. Naturalmente essa representação envolve uma grande redundância. Para reduzir essa redundância pode-se amostrar a função TWC(a,b) em determinados pontos como na grade indicada na Fig. 2.2 dando origem à Transformada *Wavelet* Discreta (TWD). A grade de amostragem não é arbitrária. Ela deve ser escolhida de modo a garantir as condições de inversibilidade da transformada como descrito em [13], [16].

2.2.2 A Transformada *Wavelet* Discreta (TWD)

A Transformada *Wavelet* Discreta de um sinal $s(t)$ é definida como

$$TWD(m,n) = \int_{-\infty}^{+\infty} s(t)\psi_{m,n}(t)dt, \tag{2.7}$$

onde

$$\psi_{m,n}(t) = a_0^{-\frac{m}{2}}\psi(a_0^{-m}t - nb_0), \qquad \psi_{0,0}(t) = \psi(t), \tag{2.8}$$

ou seja, os parâmetros a e b são discretizados nos valores $a = a_0^m$ e $b = nb_0a_0^m$, sendo a_0 e b_0 constantes que determinam os intervalos de amostragem e m, n inteiros. Tanto $s(t)$ quanto $\psi(t)$ ainda são contínuos. Para se obter eficiência computacional, considera-se $a_0 = 2$ e $b_0 = 1$, resultando em uma dilatação binária de 2^m e em uma translação diádica de 2^m n [12], como mostrado na Fig. 2.2. Assim, a Eq. 2.8 pode ser expressa como

$$\psi_{mn}(t) = 2^{-m/2}\psi\left(2^{-m}t - n\right). \tag{2.9}$$

A Fig. 2.2 mostra uma grade de amostragem diádica do plano deslocamento-escala da TW. Cada nó corresponde a uma função *wavelet* $\psi_{mn}(t)$ com escala 2^m e deslocamento $2^m n$.

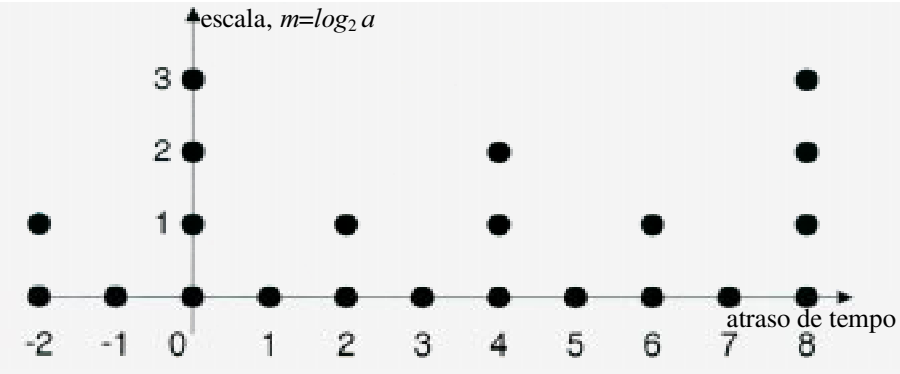


Fig. 2.2: Grade de amostragem diádica do plano deslocamento-escala da TW.

Com (*a*,*b*) discretos é ainda possível ter uma representação completa de *s(t),* desde que a taxa de amostragem seja suficientemente densa. Dependendo do tipo de $\psi(t)$ usado e da grade de amostragem de (*a*,*b*), algumas restrições são exigidas para que se tenha uma reconstrução perfeita. É interessante analisar o tipo de $\psi(t)$ e os intervalos de amostragem para (*a*,*b*) que permitem reconstrução perfeita.

A Transformada *Wavelet* Discreta Inversa é dada por

$$s(t) = c\sum_m \sum_n TWD(m,n)\psi_{mn}(t), \tag{2.10}$$

onde *c* é uma constante que depende somente de $\psi(t)$.

2.2.3 Princípios de Análise e Síntese de Multiresolução

As interpretações matemáticas para a Transformada *Wavelet* são melhores representadas quando se considera o conceito de multiresolução [17]. A idéia básica da análise (decomposição) de multiresolução (MRA – *multiresolution analysis*) é similar à decomposição

e codificação por subbandas, onde, para se obter uma codificação eficiente, um sinal é dividido em um conjunto de subbandas de freqüências [18]. Estas subbandas são então codificadas individualmente, de acordo com sua energia e importância relativa para a recuperação do sinal. Nesta seção analisa-se o conceito da decomposição de multiresolução para sinais de uma dimensão. A seguir o modelo é estendido para duas dimensões para a sua utilização em compressão de imagens.

Filtros de diferentes faixas de freqüências são usados para analisar o sinal em diferentes escalas (subbandas). Inicialmente o sinal é introduzido em um par de filtros digitais passa-baixa e passa-alta, para que sejam analisadas as baixas e altas freqüências, respectivamente. A largura de faixa destes filtros é tipicamente a metade da largura de faixa do sinal original. Daí utilizar-se a denominação de filtros de meia banda (*halfband filters*). A resolução do sinal, que é uma medida da quantidade de detalhes, é alterada pelas operações de filtragem. Como a faixa do sinal filtrado é reduzida pela metade, pode-se dizimar o conjunto de amostras filtradas, por exemplo, considerando-se apenas as amostras pares (i.e, em múltiplos pares dos instantes de amostragem) e descartando-se as amostras ímpares. A dizimação de um sinal por um fator de *n* reduz o número de amostras *n* vezes.

Esta decomposição em duas bandas pode ser recursivamente aplicada ao sinal através de um banco de filtros, para produzir uma decomposição em 2^m bandas de freqüências. A Transformada *Wavelet* Discreta (TWD) utiliza esta decomposição em bancos de filtros aplicados exclusivamente à saída do filtro passa-baixa. Os filtros passa-baixa e passa-alta são relacionados às funções de escala (*scaling function*) e de *wavelet*, respectivamente. Assim, a decomposição do sinal em diferentes subbandas de freqüências é simplesmente obtida pelas sucessivas filtragens das componentes dizimadas resultantes dos filtros passa-baixa, i.e., das alterações de escala do sinal. A Fig. 2.3 apresenta um diagrama com a seqüência de operações para a obtenção da TWD. O sinal original *s(n)* é introduzido em um par de filtros de síntese $\bar{g}(n)$ e $\bar{h}(n)$, passa-baixa e passa-alta respectivamente. Os sinais resultantes da saída dos filtros são dizimadas por um fator de 2. Estas operações constituem o primeiro nível de decomposição e podem ser expressas como

$$y_L(n) = \sum_n s(n).\bar{g}(2k-n) \qquad \text{e} \qquad y_H(n) = \sum_n s(n).\bar{h}(2k-n), \tag{2.11}$$

onde, $y_L(n)$ e $y_H(n)$ são as saídas dos filtros passa-baixa e passa-alta, respectivamente, depois da dizimação por um fator de 2 (chamados sinal de aproximação e de detalhamento). Esta operação é repetida sucessivas vezes para o sinal de aproximação, resultando nos sinais de diferentes resoluções que compõem a TWD em 3 níveis de decomposição ($y_{LLL}(n), y_{LLH}(n), y_{LH}(n)$ e $y_H(n)$). Os sinais em diferentes resoluções são agrupados dentro de uma estrutura piramidal chamada pirâmide laplaciana. Os dados da pirâmide laplaciana, como observado por Burt [19] e Crowley [20], possuem a característica de serem correlatados nos diferentes níveis de resolução. No próximo capítulo analisa-se um algoritmo que lida com esta correlação entre os níveis de decomposição.

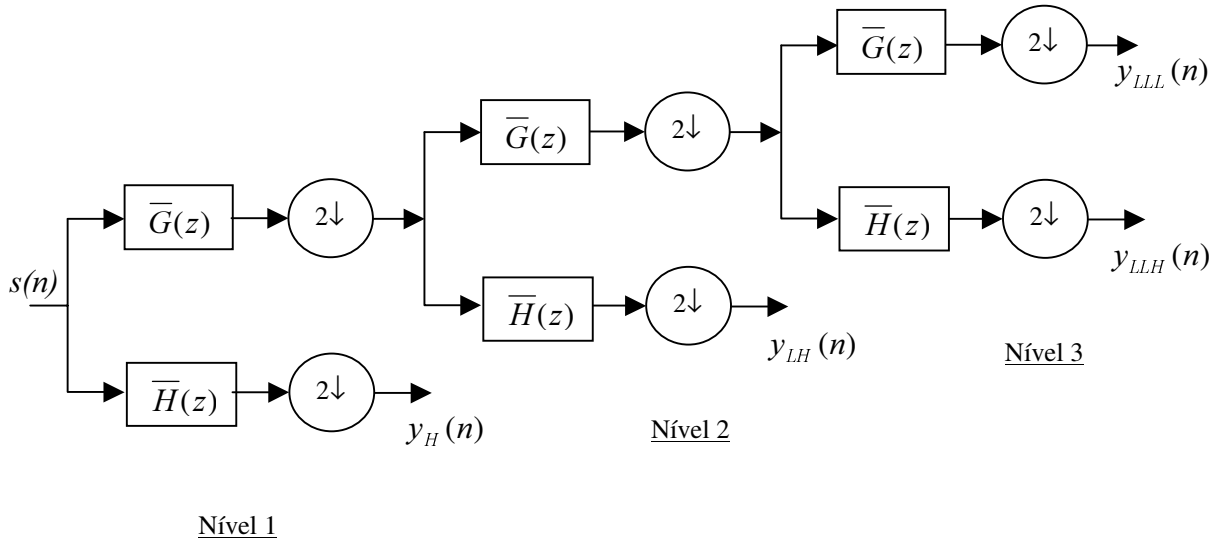


Fig. 2.3: Análise de Multiresolução (1-D) para 3 níveis de decomposição.

O processo de divisão do espectro da freqüência é graficamente mostrado na Fig. 2.4.

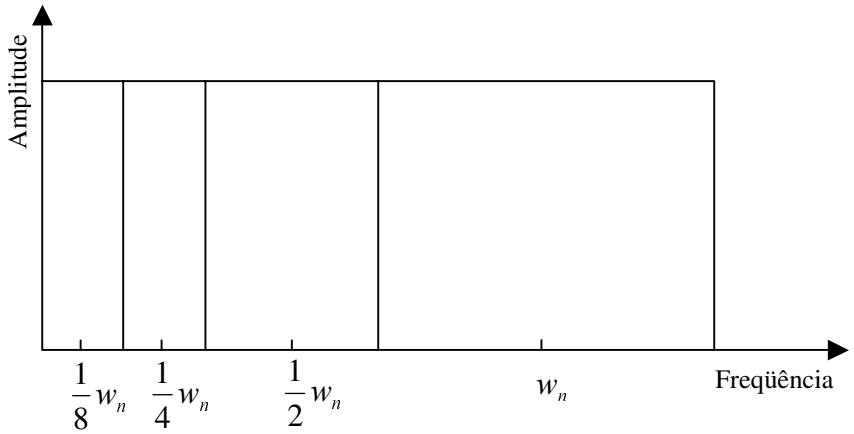


Fig. 2.4: Divisão do espectro da freqüência para 3 níveis de decomposição.

Primeiramente o sinal é dividido em uma banda de baixas freqüências e uma de altas freqüências. A seguir a banda de baixas freqüências é dividida em duas, e assim sucessivamente, dependendo do número de decomposições. Esta figura é apenas representativa das decomposições em subbandas, pois em geral, os filtros utilizados produzem bandas que se superpõem parcialmente.

Antes da transformada *wavelet* o sinal tem uma boa localização no tempo mas uma má localização na freqüência. Depois que o sinal é transformado, ele passa a ter uma boa resolução no tempo nas altas freqüências, e boa resolução em freqüência nas baixas freqüências, como mostrado na Fig. 2.5. Isto faz sentido quando o sinal a ser analisado tem componentes de altas freqüências em curtas durações e componentes de baixas freqüências em longas durações. Os sinais que são encontrados em aplicações práticas geralmente apresentam esta característica.

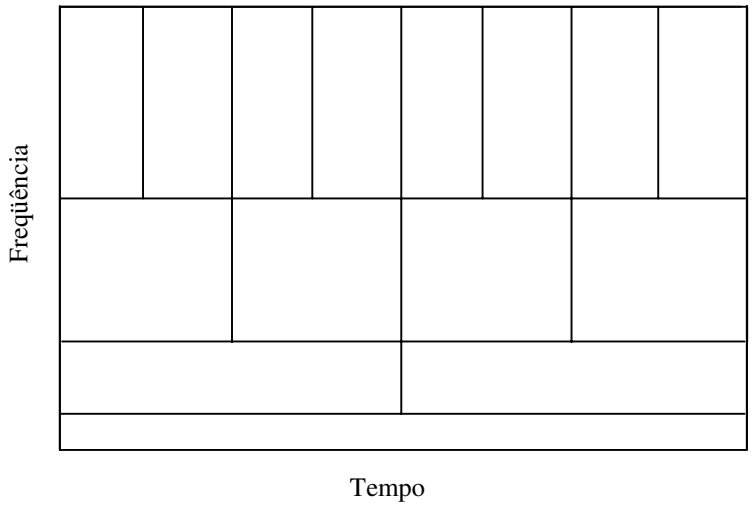


Fig. 2.5: Resolução no tempo-freqüência da TW.

O procedimento de reconstrução do sinal é seguido na ordem reversa da análise sendo a operação de dizimação substituída pela operação de interpolação. Interpolar um sinal corresponde a aumentar a taxa de amostragem pela adição de novas amostras no sinal. Esta operação é normalmente realizada pela inclusão de *n-1* amostras nulas entre as amostras originais, e pela passagem do sinal obtido por um filtro de interpolação. Interpolação de um sinal por um fator de *n* aumenta o número de amostras *n* vezes.

Para a reconstrução do sinal $s(n)$ da Fig.2.3, os sinais em cada nível de decomposição são interpolados por um fator de 2, introduzidos nos filtros de síntese $g(n)$ e $h(n)$ (passa-baixa e passa-alta, respectivamente) e adicionados. Este processo é repetido para o mesmo número de

decomposições que foram realizadas no processo de análise. O número total de pixels nesta nova representação é igual ao número de pixels da imagem original. Isto ocorre devido à ortogonalidade da representação. Portanto, a fórmula de reconstrução se torna (para cada camada)

$$\hat{s}(n) = \sum_{k=-\infty}^{+\infty} \left[y_L(n).g(-n+2k) \right] + \left[y_H(n).h(-n+2k) \right] \tag{2.12}$$

O sinal reconstruído $\hat{s}(n)$ deve ser uma aproximação do sinal original $s(n)$ para que este processo seja eficiente. Portanto, a escolha dos filtros de análise e síntese deve ser feita de uma maneira cuidadosa, pois eles podem afetar significativamente a qualidade do sinal analisado. Na próxima seção estudam-se bancos de filtros que possibilitam a reconstrução perfeita do sinal decomposto.

Uma das aplicações usuais da análise de multiresolução é a compressão de sinais de imagens. Assim, é necessário definir a transformada *wavelet* para sinais bidimensionais. A transformada *wavelet* bidimensional pode ser calculada como uma extensão separável do algoritmo de decomposição unidimensional.

O algoritmo de decomposição bidimensional é ilustrado pelo diagrama de blocos da Fig. 2.6. Primeiramente as linhas da imagem original são filtradas com um par de filtros unidimensionais passa-baixa e passa-alta, $\overline{g}(n)$ e $\overline{h}(n)$ respectivamente. A seguir, os sinais resultantes do processo de filtragem são dizimados por um fator de 2. O mesmo processo é então repetido para as colunas da imagem.

Após o primeiro estágio de decomposição são geradas 4 subbandas de freqüência. A subbanda que contém a aproximação do sinal em baixas freqüências é denominada $y^1{}_{LL}(n_1, n_2)$. É nesta subbanda que está concentrada tipicamente, a maior parte da informação da imagem. Ela é uma versão suavizada e dizimada da imagem original, e exibe alta correlação entre os coeficientes. As subbandas que contêm as altas freqüências na vertical, horizontal e diagonal são chamadas $y^1{}_{LH}(n_1, n_2)$, $y^1{}_{HL}(n_1, n_2)$ e $y^1{}_{HH}(n_1, n_2)$ respectivamente. Estas subbandas contêm os detalhes da imagem, e apresentam regiões de coeficientes aproximadamente nulos correspondentes as regiões de baixa atividade espacial da imagem original.

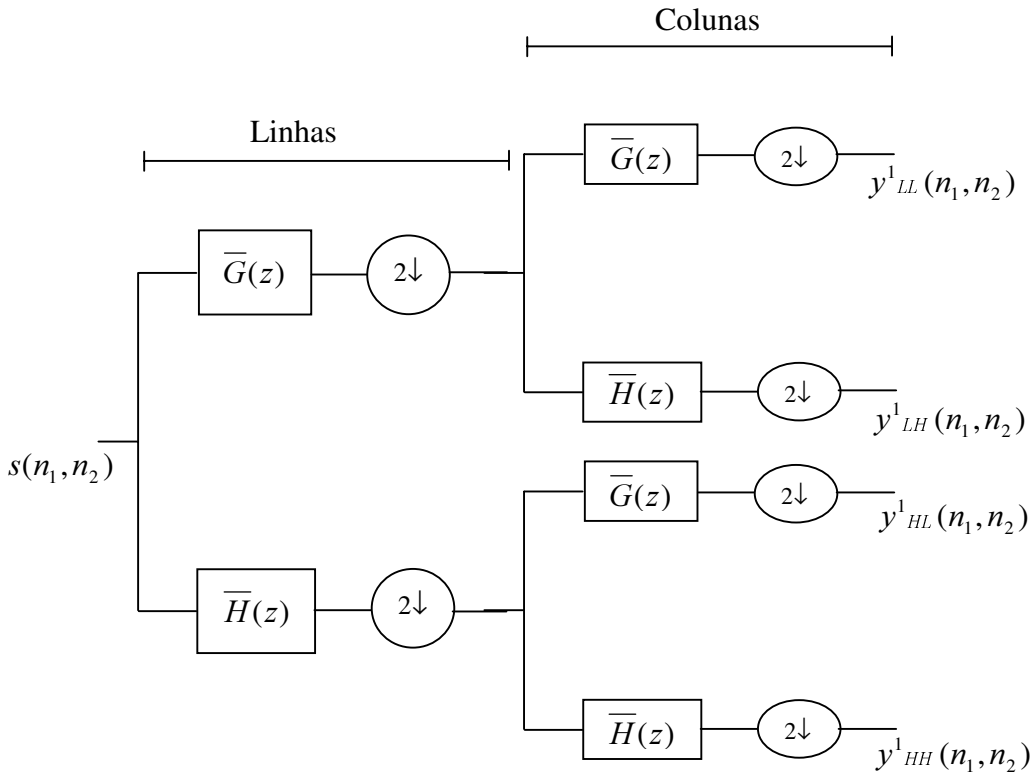


Fig. 2.6: Análise de Multiresolução Bidimensional
para um nível de decomposição.

O próximo estágio de decomposição é realizado sobre a subbanda $y^1\ (n_1,n_2)$, onde serão geradas as subbandas $y^2\ (n_1,n_2)$, $y^2{}_H\ (n_1,n_2)$, $y^2{}_{HL}(n_1,n_2)$ e $y^2{}_{HH}(n_1,n_2)$. Este processo pode ser repetido sucessivas vezes. Para *M* níveis de decomposição, serão geradas *3M* subbandas de detalhamento, $y^m(n_1,n_2)$, $1 \le m \le M$, e uma aproximação final em baixas freqüências, $y^M{}_{LL}(n_1,n_2)$. A Fig. 2.7 ilustra este processo para três estágios de decomposição.

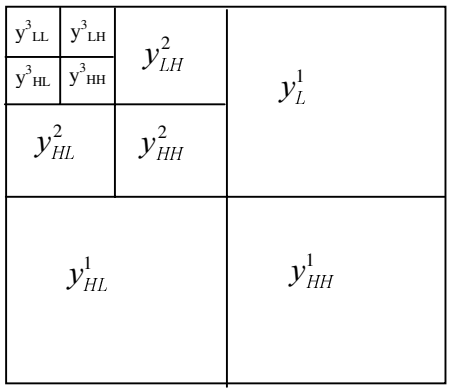


Fig. 2.7: Análise de multiresolução bidimensional
para três estágios de decomposição.

Todo estágio da decomposição em subbandas ou análise MRA é implementado através da filtragem das linhas e colunas pelos filtros de análise $\overline{g}(n)$ e $\overline{h}(n)$. Após estas convoluções, as linhas e colunas da imagem são dizimadas. No entanto, sabe-se que a convolução linear de um sinal discreto de comprimento N_1 com um filtro qualquer de comprimento N_2 produz uma saída de comprimento N_1+N_2-1. Para haver conservação do número total de amostras, as convoluções devem manter o comprimento N_1 original das linhas e colunas. Assim depois de dizimada, cada linha ou coluna deverá ter um comprimento de $N_1/2$, onde N_1 é par. Para alcançar este resultado, há duas alternativas [21]: uma é a utilização da convolução circular e a outra é a utilização da extensão simétrica da linha ou coluna a ser filtrada. Ambas alternativas satisfazem o critério de reconstrução perfeita, que será visto posteriormente, e exibem a mesma complexidade computacional.

O algoritmo de reconstrução unidimensional também pode ser estendido para o caso bidimensional. Em cada nível a imagem decomposta é reconstruída a partir das subimagens $y^m{}_{LL}(n_1,n_2)$, $y^m{}_{LH}(n_1,n_2)$, $y^m{}_{HL}(n_1,n_2)$ e $y^m{}_{HH}(n_1,n_2)$. Este algoritmo é ilustrado pelo diagrama de blocos da Fig. 2.8.

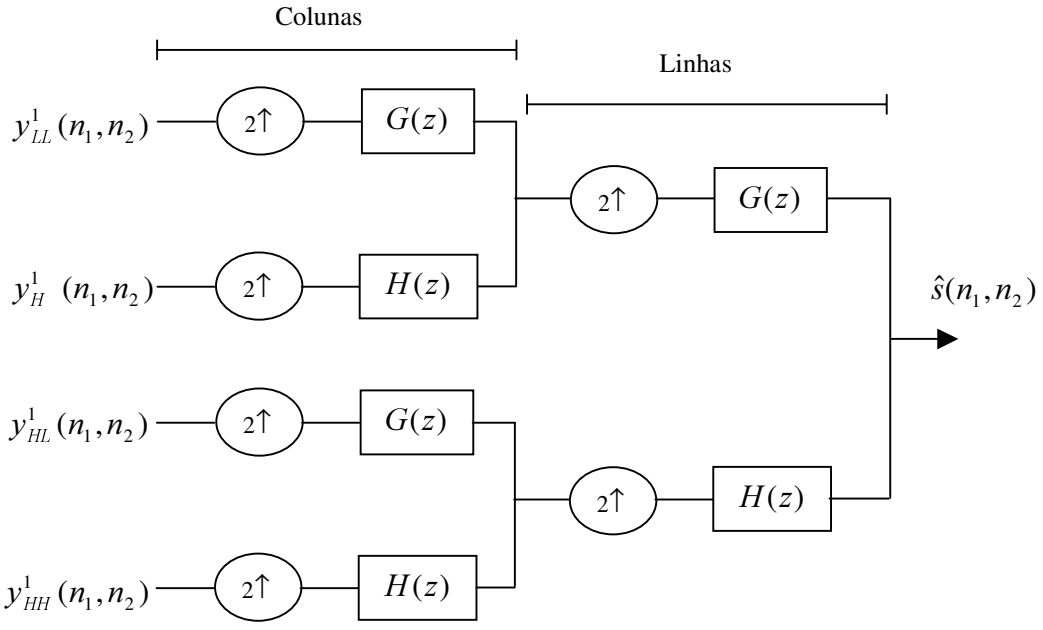


Fig. 2.8: Síntese de Multiresolução Bidimensional
para um nível de decomposição.

Entre as colunas das subimagens $y^1{}_{LL}(n_1,n_2)$, $y^1{}_{LH}(n_1,n_2)$, $y^1{}_{HL}(n_1,n_2)$ e $y^1{}_{HH}(n_1,n_2)$ é adicionada uma coluna de zeros e então as linhas são covoluídas com um filtro unidimensional. As subbandas $y^1{}_{LL}(n_1,n_2)$ e $y^1{}_{LH}(n_1,n_2)$ são somadas formando uma nova subimagem, o mesmo procedimento é seguido para as subbandas $y^1{}_{HL}(n_1,n_2)$ e $y^1{}_{HH}(n_1,n_2)$, como mostrado na Fig. 2.8. A seguir adiciona-se uma linha de zeros entre as linhas das subimagens resultantes e faz-se a convolução das colunas com um filtro unidimensional. As subimagens resultantes do processo anterior são somadas formando a imagem reconstruída $\hat{s}(n_1,n_2)$. Os filtros usados na reconstrução são os filtros $g(n)$ e $h(n)$ (filtros passa-baixa e passa-alta, respectivamente).

2.2.4 Filtros para Reconstrução Perfeita

Nesta seção investigam-se as especificações dos filtros *g(l)* e *h(l)* para que o requisito de reconstrução perfeita seja satisfeito. A transformada Z do sinal resultante $\hat{s}(n)$ da Eq. (2.12) é

$$\hat{S}(z) = T(z)S(z) + E(z)S(-z). \tag{2.13}$$

A transformada *T(z)* que aparece multiplicando a transformada do sinal original é dada por

$$T(z) = \overline{G}(z)G(z) + \overline{H}(z)H(z). \tag{2.14}$$

O termo *E(z)*, que multiplica *S(-z)*, é dado por

$$E(z) = \overline{G}(-z)G(z) + \overline{H}(-z)H(z). \tag{2.15}$$

Para que haja reconstrução perfeita $\hat{S}(z)$ deve ser uma cópia, possivelmente atenuada ou atrasada, de $S(z)$. Portanto, T(z) deve ser igual a

$$T(z) = \alpha Z^{-D}, \tag{2.16}$$

onde "*D*" é um atraso inteiro e "α" é um fator de atenuação. Sem grande perda de generalidade, considera-se $\alpha = 1$. Além disso, o segundo termo na Eq. 2.13 deve ser nulo, o que implica em

$$E(z) = 0\,. \tag{2.17}$$

Isto equivale a anular na saída o termo *S(-z)* que é chamado termo de *aliasing*, permanecendo somente uma réplica de *s(n)* atrasada de "*D*" unidades. O termo *S(-z)* equivale a $S(\omega + \pi)$ no domínio da transformada de *Fourier*. Caso este termo não seja anulado na Eq. 2.13, ele causará distorção.

Para satisfazer a primeira condição para reconstrução perfeita, Eq. 2.16, é suficiente fazer

$$H(z) = -z^{-(p-1)}G(-z^{-1}). \tag{2.18}$$

Para que a segunda condição de reconstrução perfeita seja satisfeita, Eq. 2.17, pode-se construir os filtros de síntese $G(z)$ e $H(z)$ de modo que

$$G(z) = -\overline{H}(-z)\,, \quad \text{e} \quad H(z) = \overline{G}(-z)\,. \tag{2.19}$$

Assim,

$$T(z) = G(z)H(-z) - G(-z)H(z). \tag{2.20}$$

Recapitulando, as condições para reconstrução perfeita são,

$$T(z) = \alpha Z^{-} \quad \text{e} \quad E(z) = 0\,. \tag{2.21}$$

Uma solução para as Eqs. 2.18 e 2.19 dos bancos de filtros de análise e de síntese é a chamada solução paraunitária. Neste caso, ambos os filtros *h(l)* e *g(l)* são do tipo FIR e possuem a mesma ordem *p* par. Rescreve se então *T(z)* como,

$$T(z) = z^{-(p-1)}[R(z) + R(-z)] \tag{2.22}$$

onde,

$$R(z) = G(z)G(z^{-1}). \tag{2.23}$$

Para o filtro passa-baixa, $G(z)$, tem-se,

$$G(z) = \sum_{l=0}^{p-1} g(l) z^{-1}. \tag{2.24}$$

Levando-se em conta

$$s(n) = \sqrt{2} \sum_{l=0}^{p-1} g(l) c^1_{(n-l)/2} + \sqrt{2} \sum_{l=0}^{p-1} h(l) d^1_{(n-l)/2}, \tag{2.25}$$

a condição para reconstrução perfeita é

$$\begin{aligned} 1 = R(z) + R(-z) = 2 \sum_{l} g^2(l) + \sum_{i=0}^{p-1} \sum_{k=0, k \neq i}^{p-1} g(i) g(k) z^{k-i} \\ + \sum_{l=0}^{p-1} \sum_{m=0, m \neq l}^{p-1} g(l) g(m) z^{m-l} (-1)^{m-l}. \end{aligned} \tag{2.26}$$

Os últimos dois termos da Eq. 2.26 somam "0" se ambos "*k - i*" e "*m - l*" forem iguais e ímpares, o que resulta em

$$1 = 2 \sum_{l=0}^{p-1} g^2(l) + 2 \sum_{k-ipar, \neq 0} g(i) g(k) z^{k-i}. \tag{2.27}$$

Devido a condição de ortogonalidade de $\phi(t)$, tem-se

$$\sum_{l=0}^{p-1} g^2(l) = \frac{1}{2}, \tag{2.28}$$

assim, obtém-se a condição de reconstrução perfeita a partir da Eq. 2.27

$$\sum_{i=0}^{p-1} g(i) g(i+2k) = \begin{cases} 0, & k = 1,2,.... \\ 1/2, & k = 0. \end{cases} \tag{2.29}$$

Adicionalmente, segue da Eq. 2.18 que

$$h(l) = (-1)^l g(p-1-l), \tag{2.30}$$

ou seja,

$$\sum_{l=0}^{p-1} h(l)g(l) = 0. \tag{2.31}$$

As respostas impulsivas e os espectros dos filtros são ortogonais entre si. A propriedade (2.29) também é verificada para *h(l)*, $\bar{g}(l)$ e $\bar{h}(l)$. Como *g(l)* determina todos os outros filtros, toda a análise de multiresolução está amarrada ao projeto do filtro *g(l)*.

Os filtros na Eq. 2.18 possuem resposta espectral em amplitude que satisfazem

$$\begin{aligned} &| G(w) | = | H(\pi - w) | = | H(w + \pi) |, \\ &| H(w) |^2 + | H(w + \pi) |^2 = 1, \\ &| G(w) |^2 + | G(w + \pi) |^2 = 1. \end{aligned} \tag{2.31}$$

Tais filtros são chamados filtros espelho em quadratura (QMF – *Quadrature Mirror Filters*), já que apresentam respostas refletidas com respeito à freqüência π/2.

Na Fig. 2.9 é mostrado o gráfico idealizado, utilizando-se a normalização G(0) = 1. Com esta normalização,

$$| H(w) |^2 + | G(w) |^2 = 1, \tag{2.32}$$

isto é, os filtros tornam-se complementares em potência [18]. Além disto, a Eq. 2.32 acarreta em $H(0) = 0$. Esta afirmativa equivale no domínio Z a $H(z = e^{j\omega}) |_{\omega=0} = H(z = 1) = 0$, ou seja, *H(z)* deve possuir pelo menos um zero em *z* = 1. A partir da Eq. 2.18 conclui-se também que *G(z =-1)=0*, ou seja, *G(z)* deve possuir pelo menos um zero em *z* = -1.

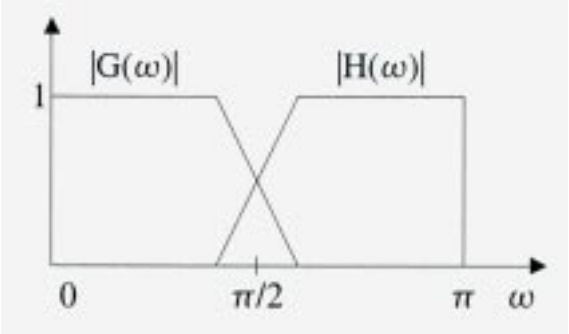


Fig. 2.9: Filtros QMF.

Filtros FIR paraunitários, apesar de possuírem algumas propriedades valiosas e gerarem *wavelets* ortonormais de suporte compacto [22], não possuem fase linear, exceto para o caso trivial em que p=2, que leva à *wavelet* de *Haar* [23]. Seus coeficientes e por conseqüência, a *wavelet* $\psi(t)$ que estes filtros geram, não são simétricos nem anti-simétricos. No entanto, a propriedade de fase linear é desejável em aplicações como codificação de imagens, onde distorções devido à não linearidade de fase são facilmente notadas [2]. Assim, filtros que possuem fase linear são usualmente empregados em compressão de imagens. A seguir, é mostrado como contornar este problema, lançando-se mão de suporte compacto para a *wavelet* $\psi(t)$.

2.2.5 *Wavelets* Biortogonais

Apesar da ortogonalidade apresentar muitas vantagens, em algumas aplicações, como na compressão de imagens, é importante que a transformada utilizada apresente ainda certas características como regularidade, simetria e suporte compacto [1].

A regularidade das *wavelets* está relacionada com o tipo de função que a transformada pode representar. Quanto maior a regularidade, maior o número de sinais possíveis de serem representados com maior fidelidade.

Por outro lado, a simetria das *wavelets* exige a utilização de filtros de fase linear. Tais filtros permitem cascateamento de bancos de filtros sem a necessidade de compensação de fase. Além disso, filtros de fase linear não propagam os erros de cada estágio [1].

Por último, o suporte compacto exige que os filtros apresentem resposta impulsiva de duração finita (filtros FIR) e proporciona algoritmos eficientes. Naturalmente, os filtros utilizados no esquema de análise/síntese devem garantir a reconstrução perfeita do sinal, ou seja, nenhuma informação deve ser perdida.

Como foi visto anteriormente, a única *wavelet* ortogonal que obedece a todas estas exigências é a *wavelet* de *Haar* [11]. Esta *wavelet*, entretanto, não é contínua e não é adequada para aplicações práticas. Para contornar este problema, relaxa-se a condição de ortogonalidade, dando lugar a condição de biortogonalidade, que é descrita a seguir. Existem *wavelets* biortogonais que atendem a todas as exigências citadas acima. Além disso, os filtros utilizados para a sua implementação são bem mais flexíveis e fáceis de projetar.

Na família biortogonal, a seguinte relação é satisfeita

$$\langle \psi_{mn}(t), \tilde{\psi}_{kl}(t) \rangle = \delta[m-k]\delta[n-l] \tag{2.33}$$

onde $\tilde{\psi}_{kl}(t)$ denota a função dual. A dualidade das *wavelets* ψ e $\tilde{\psi}$ é relatada no seguinte sentido

$$\int \psi_{mn}(t)\tilde{\psi}_{kl}(t)dt = 0 \tag{2.34}$$

sabendo que $m \neq n$ e $k \neq l$.

Esta propriedade é importante pois as propriedades úteis para o processo de análise podem ser concentradas na função ψ (como por exemplo, oscilação, momento zero, etc) enquanto que as propriedades interessantes para síntese (regularidade) são fixadas para a função $\tilde{\psi}$.

Se a família é completa em um dado espaço tal como $\mathbf{L}^2(\Re)$, então qualquer função deste espaço de funções pode ser escrita como

$$s(t) = \sum_m \sum_n \langle \psi_{mn}, s \rangle \tilde{\psi}_{mn}(t) \tag{2.35}$$

$$= \sum_m \sum_n \langle \tilde{\psi}_{mn}, s \rangle \psi_{mn}(t), \tag{2.36}$$

sabendo-se que ψ e $\tilde{\psi}$ são duais.

A partir das Eqs. 2.14 a 2.21 obteve-se as condições para reconstrução perfeita. A Fig. 2.10 esboça os filtros biortogonais e suas posições para se obter a reconstrução perfeita da imagem.

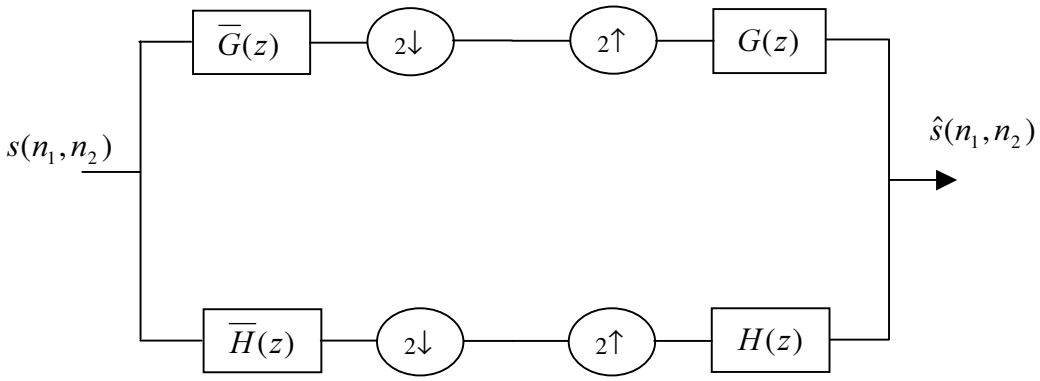


Fig. 2.10: Filtros biortogonais para reconstrução perfeita.

A partir da Eq. 2.19 tem-se

$$G(z)H(z) = -H(z)\overline{H}(-z), \tag{2.37}$$

e

$$H(z)G(z) = G(z)\overline{G}(-z). \tag{2.38}$$

Os produtos presentes em ambos os membros das igualdades expressas nas Eqs. 2.37 e 2.38 devem ter coeficientes nulos para potências ímpares de z. Portanto, tem-se

$$\overline{G}(z) = \sum_{l=0}^{p-1} \overline{g}(l) z^{-l} \quad , \quad G(z) = \sum_{l=0}^{q-1} g(l) z^{-1} \tag{2.39}$$

e

$$\overline{H}(z) = \sum_{l=0}^{q-1} \overline{h}(l) z^{-1} \quad , \quad H(z) = \sum_{l=0}^{p-1} h(l) z^{-l}, \tag{2.40}$$

onde os filtros passa-baixa ou passa-alta de análise e de síntese podem ter agora, ordens p e q diferentes. Pode-se afirmar que

$$\sum_{l=0}^{q-1} g(l)[\overline{h}(l+k) + \overline{h}(l-k)] = 0, \quad k \text{ ímpar}, \tag{2.41}$$

$$\sum_{l=0}^{p-1} \overline{g}(l)[h(l+k) + h(l-k)] = 0, \quad k \text{ ímpar}. \tag{2.42}$$

As Eqs. 2.37 a 2.42 definem um sistema de filtros biortogonais. Procedimentos de projeto podem ser encontrados em [24]. Os filtros biortogonais possuem fase linear de modo que: (i) $\overline{G}(z)$ e $\overline{H}(z)$ podem ambos ser simétricos de ordem ímpar diferindo por múltiplos ímpares de dois, ou (ii) $\overline{G}(z)$ pode ser simétrico e $\overline{H}(z)$ anti-simétrico, ou vice-versa com ambas as ordens pares iguais ou diferindo por múltiplos pares de dois. Neste caso, as Eqs. 2.29 e 2.30 não são mais válidas para os filtros de análise e síntese. Conseqüentemente, a TWD e a TWD Inversa necessitam de *wavelets* distintas.

As funções de escala e *wavelet* dos filtros de análise são, respectivamente,

$$\phi(t) = 2\sum_{l=0}^{p-1} \overline{g}(l)\phi(2t-l) \tag{2.43}$$

e

$$\psi(t) = 2\sum_{l=0}^{q-1} \overline{h}(l)\phi(2t-l)\,. \tag{2.44}$$

As funções de escala e *wavelet* para a análise e síntese são duais, ou seja,

$$\tilde{\phi}(t) = 2\sum_{l=0}^{p-1} g(l)\tilde{\phi}(2t-l) \tag{2.45}$$

e

$$\tilde{\psi}(t) = 2\sum_{l=0}^{-1} h(l)\tilde{\phi}(2t-l). \tag{2.46}$$

A representação para o sinal contínuo *s(t)* é dada por

$$s(t) = \sum_m \sum_n \left\langle s(t), \psi_{mn}(t) \right\rangle \tilde{\psi}(t). \tag{2.47}$$

As seguintes condições de ortogonalidade valem para as funções de escala e *wavelet* apresentadas nas Eqs. 2.43 a 2.46

$$\begin{aligned} &\left\langle \phi(t-k), \tilde{\phi}(t-l) \right\rangle = \delta(k-l), \\ &\left\langle \psi(t-k), \tilde{\psi}(t-l) \right\rangle = \delta(k-l), \\ &\left\langle \phi(t-k), \tilde{\psi}(t-l) \right\rangle = 0, \\ &\left\langle \psi(t-k), \tilde{\phi}(t-l) \right\rangle = 0. \end{aligned} \tag{2.48}$$

Em particular,

$$\left\langle \psi_{mn}(t), \tilde{\psi}_{jk}(t) \right\rangle = \delta(m-j)\delta(n-k). \tag{2.49}$$

Em resumo, os filtros biortogonais conferem a cada estágio da decomposição de multiresolução a propriedade de fase linear, regularidade, simetria e suporte compacto, que são interessantes ao se trabalhar no processamento de imagens.

2.3 A Quantização

Seja x uma grandeza escalar contínua representando a intensidade de um pixel (um elemento de uma imagem). A quantização de x consiste na limitação da precisão com que a grandeza é representada. Para isso, a faixa dinâmica de grandeza é repetida em intervalos de quantização. A representação de um valor em um determinado intervalo é feita por um ponto previamente selecionado, tipicamente o ponto médio do intervalo. Para representar x com um número finito de bits, um número finito de intervalos de quantização é normalmente utilizado. Neste trabalho, considera-se que L níveis são utilizados para representar x. Se cada pixel ou coeficiente da imagem é quantizado de forma independente, o procedimento é denominado quantização escalar. Se dois ou mais coeficientes forem quantizados conjuntamente, o procedimento é denominado quantização vetorial [5], [25].

2.3.1 Quantização Escalar

Existem várias formas de quantização escalar. Dentre elas, destacam-se a quantização uniforme e a quantização de Llyod-Max [26], [27]. Uma quantização escalar, em geral, pode ser interpretada como um mapeamento Q[.] de um subconjunto x da reta Real em um conjunto discreto de pontos em $\Re$. Um exemplo do mapeamento de um quantizador escalar é apresentado na Fig. 2.11.

O mapeamento Q[.] é definido através de uma partição do conjunto X em intervalos não sobrepostos $X_i = [x_{i-1}, x_i)$. Os limites deste intervalo, x_{i-1} e x_i, são denominados limiares de decisão. Os intervalos devem satisfazer às seguintes condições:

$$\bigcup_{i=0}^{L-1} X_i = X \quad , \quad X_i \bigcap X_j = \phi. \tag{2.50}$$

A cada intervalo X_i é associado um nível de representação y_i. Portanto, se o conjunto X for dividido em L intervalos, haverá L níveis de representação. Os valores do eixo vertical das transições do gráfico da Fig. 2.11 correspondem aos níveis de representação $\{y_i,\ i = 0,1,...,L-1\}$ do quantizador. Os valores associados às transições do gráfico no eixo horizontal correspondem aos limiares de decisão x_i.

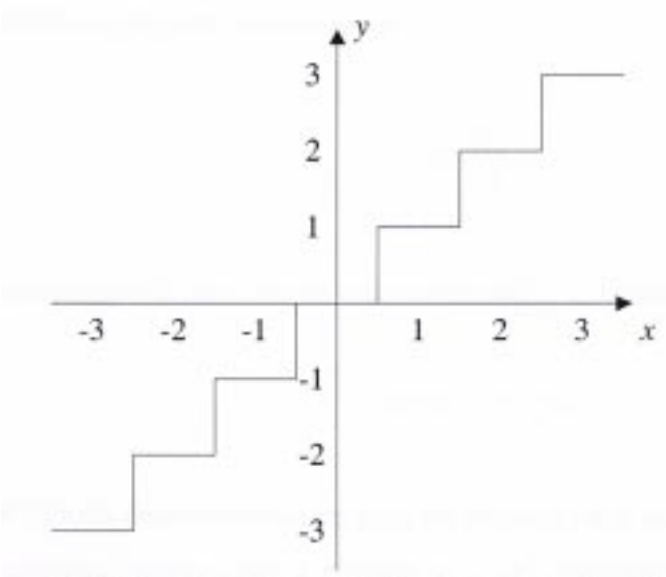


Fig. 2.11 : Quantização uniforme.

Seja $\hat{x}$ o valor quantizado de x. O quantizador pode ser representado por

$$\hat{x} = Q(x) = y_i \quad \text{se} \quad x \in X_i = [x_{i-1}, x_i] \qquad i = 0,1,..., \text{L-1} \tag{2.51}$$

A partir da Eq. 2.51 tem-se que se x estiver entre x_{i-1} e x_i ele é mapeado para o nível de reconstrução y_i.

O desempenho de um quantizador é medido através do erro introduzido pela quantização. Este erro é denominado ruído de quantização e é definido por

$$q[n] = \hat{x}[n] - x[n] \tag{2.52}$$

onde x e $\hat{x}$ são, respectivamente, a entrada e a saída do quantizador. O valor esperado de q^2, determinado erro quadrático médio, é freqüentemente adotado como uma medida da distorção introduzida pelo quantizador.

Os níveis de representação e limiares de decisão são freqüentemente determinados através da minimização da medida adotada de distorção.

Voltando ao erro quadrático médio (MSE), tem-se:

$$MSE = E(x-\hat{x})^2 = \sum_{i=0}^{L-1} \int_{x_{i-1}}^{x_i} (x-y_i)^2 \, p_x(x_0) dx_0 \tag{2.53}$$

Quando $p_x(x)$ é uniforme, a quantização ótima possui níveis de representação e decisão uniformemente espaçados e definidos através das expressões

$$x_{i-1} - x_i = \Delta \quad \text{e} \quad y_i = \frac{(x_{i-1} + x_i)}{2}, \qquad 1 \le i \le L \tag{2.54}$$

onde Δ é o tamanho do intervalo ou passo de quantização, i.e., o espaçamento entre dois níveis de representação ou de decisão consecutivos. Para este tipo de quantização, denominada quantização uniforme, o valor do MSE é aproximadamente igual a $\Delta^2/12$ [5].

A quantização uniforme é muito simples e fácil de implementar. Os parâmetros do projeto são os passos de quantização (Δ) e o número de intervalos (L), ou os limites da faixa dinâmica do sinal. Estes limites são, freqüentemente, escolhidos como múltiplos do desvio padrão σ_x da função densidade de probabilidade do sinal de entrada. Tipicamente, para sinais de média nula, o limite de faixa dinâmica são simétricos em relação a origem. Sejam *a* e *–a* estes limites. Então, $\Delta = \frac{2a}{N}$ e $N = 2^R$, onde R é o número de bits do quantizador. Então, a distorção para este tipo de quantização é dada por

$$D = \Delta^2/12 = \frac{(2a)^2}{12.N^2} = \frac{a^2}{3N^2} = \frac{a^2}{3} 2^{-2R} \tag{2.55}$$

Se a densidade de probabilidade do sinal não é uniforme, a quantização ótima também não será uniforme. Um quantizador ótimo será aquele que minimizar (2.53) para um número fixo de níveis. A minimização da Eq. 2.53 foi investigada por Lloyd e Max [26], [27], que obtiveram as seguintes expressões para os níveis de decisão e representação

$$x_{k,opt} = \frac{1}{2}(y_{k,opt} + y_{k-1,opt}), \tag{2.56}$$

$$x_{0,opt} = -\infty \quad ; \quad x_{L,opt} = \infty, \tag{2.57}$$

$$y_{k,opt} = \frac{\int_{x_{k,opt}}^{x_{k+1,opt}} x.p_x(x)dx}{\int_{x_{k,opt}}^{x_{k+opt,opt}} p_x(x)dx}, \qquad \text{para } k = 1,2,...,L\text{-}1. \qquad (2.58)$$

A primeira expressão indica que os limiares de decisão devem estar no ponto médio entre os níveis de representação vizinhos. A última mostra que os níveis de representação ótimos são os centróides da função densidade de probabilidade nos intervalos apropriados. Estas expressões podem ser utilizadas de forma iterativa para se obter os níveis de representação e decisão ótimos. Este quantizador escalar ótimo é conhecido como quantizador de Lloyd-Max.

2.3.2 Quantização vetorial

O conceito da quantização vetorial é uma generalização da quantização escalar. O quantizador vetorial usa a dependência entre N amostras consecutivas dividindo um espaço N dimensional em células. A divisão de células é feita de forma a minimizar o erro médio de quantização (distorção). O quantizador vetorial Q[.] mapeia um vetor $x \in \Re^N$ em um conjunto finito $\mathbf{Y} = [y_0, y_1,..., y_{L-1} \mid y_i \in \Re^N]$ de vetores de representação, ou seja

$$Q : \Re^N \rightarrow \mathbf{Y}. \qquad (2.59)$$

Em uma quantização vetorial a coleção de níveis de representação é uma coleção de vetores que constituem o dicionário do código (*codebook*). As células C_i correspondem às regiões de decisão e podem ser interpretadas como polígonos sólidos no espaço N-dimensional $\Re^N$. No caso escalar, é fácil testar se uma amostra do sinal pertence ou não ao intervalo. No caso da quantização vetorial, uma abordagem indireta é necessária, onde é utilizado um critério de fidelidade ou uma medida de distorção. Assim, $x \in C_i$ se a distorção de x em relação a y_i for inferior ou igual à distorção obtida em relação a qualquer outro vetor de representação y_i para $j \neq i$. Portanto,

$$Q[x] = y_i \Leftrightarrow d(x,y_i) \leq d(x,y_j), \qquad j = 1,2,..., L\text{-}1. \qquad (2.60)$$

Quando a melhor aproximação y_i é encontrada, o índice *i* é codificado como a representação eficiente do vetor. O receptor pode reconstruir y_i simplesmente procurando a representação do vetor indexado por *i* no dicionário. A taxa de bits por amostra deste esquema é igual a $\frac{1}{N}\log_2 L$, quando se utilizam L vetores de representação N dimensional.

Apesar de apresentar resultados superiores aos da quantização escalar, a quantização vetorial implica em um grande aumento na complexidade computacional do sistema. A sua implementação exige a construção de um dicionário, o que pode ser bastante oneroso. Maiores detalhes desta técnica podem ser encontrados em [25]. Neste trabalho, utiliza-se a quantização escalar aplicada a codificadores de transformadas *wavelets* por uma questão de simplicidade.

2.4 A Codificação de Entropia

O último passo da compressão de imagens por transformadas é a codificação de entropia. Esta é uma etapa opcional que envolve um processo reversível e não envolve perda de informação. Após a quantização, os coeficientes assumem valores em um conjunto finito $\{a_i\}$. A idéia da codificação de entropia é obter um mapeamento *M* reversível para uma seqüência de bits tal que a média de bits por símbolo seja minimizada. Um exemplo é o código de *Morse*, o qual associa seqüências curtas (palavras de código) para as letras que aparecem com mais freqüência na língua inglesa enquanto reserva palavras longas para as menos freqüentes. Os parâmetros disponíveis para a busca deste mapeamento *M* são as probabilidades de ocorrência dos símbolos a_i, p(a_i). Se a variável quantizada é estacionária, as probabilidades são fixas e um mapeamento como o do código de Huffman [6] pode ser usado. Este código apresenta comprimento variável e é o código que minimiza o comprimento médio das palavras dentre todos os códigos instantâneos existentes. Os códigos instantâneos são aqueles em que nenhuma palavra é prefixa de outra [28]. Se as probabilidades variam durante a codificação, um mapeamento mais sofisticado como a codificação aritmética adaptativa tende a apresentar melhores resultados [29]. É interessante ressaltar que como a codificação de entropia não apresenta perda de informação, pode-se associar duas ou mais formas desta codificação após a quantização.

A entropia [H_a] da variável aleatória associada aos símbolos a_i, que é o limitante inferior do comprimento médio das palavras de código [6], é dada por

$$H_a = -\sum_{i=0}^{M-1} p(a_i)\log_2 p(a_i), \tag{2.61}$$

onde $p(a_i)$ é a probabilidade de ocorrência do nível de quantização i . Este limitante também se aplica ao comprimento médio de mensagens geradas por uma fonte caracterizada por variáveis aleatórias independentes e identicamente distribuídas segundo $p(a_i)$.

Pode-se mostrar que $0 \leq \mathrm{H}_a \leq \log_2 L$, onde L é o número de níveis de quantização. Portanto, a entropia é teoricamente a mínima taxa de bits média possível requerida para se codificar uma mensagem, ou ainda, a quantidade média de informação que uma mensagem contém. Assim, na codificação de fonte, o objetivo é obter uma taxa média de bits tão próxima quanto possível da entropia da fonte.

2.4.1 A Codificação de Huffman

O modelo do codificador de Huffman é melhor entendido como uma árvore binária. A palavra código é formada por uma seqüência de "0's" e "1's" que vai da "raiz" até a "folha" correspondente ao símbolo desejado. O codificador de Huffman começa com uma lista das probabilidades dos símbolos, como representado na Fig. 2.12.

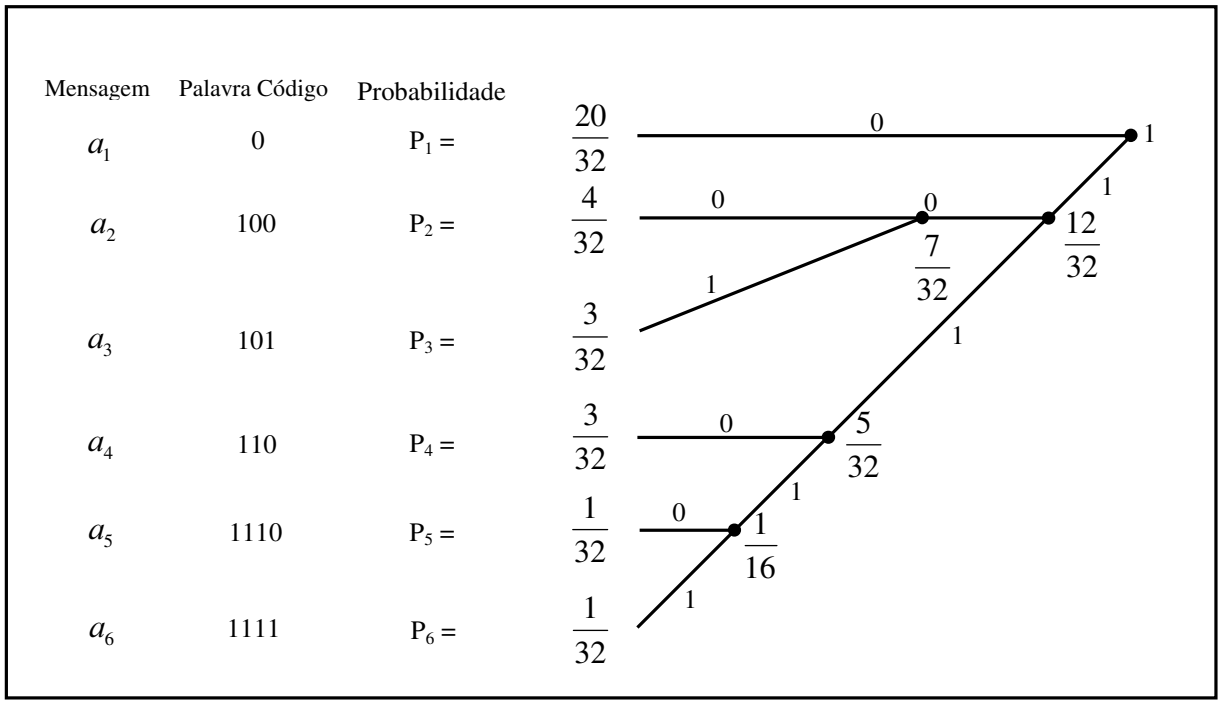


Fig. 2.12: Código de Huffman derivado de uma árvore binária.

Em seguida as probabilidades dos dois símbolos menos prováveis são somadas para formar um novo nó e as probabilidades resultantes são reordenadas. Este processo se repete até que sobre apenas uma mensagem com probabilidade 1, como mostrado na Fig. 2.12. As palavras códigos podem agora ser lidas ao longo dos "braços" da árvore binária. A codificação e decodificação das mensagens são realizadas através de consultas à árvore, que pode ser representada na forma de uma tabela.

No exemplo dado na Fig. 2.12, a entropia é dada por H = 1,752 bits/mensagem e a taxa de bits média alcançada pelo codificador de Huffman é 1,813 bits/mensagem, que corresponde a uma taxa de bits próxima da entropia.

Após o sinal de entrada passar por estes três processos (Transformada *Wavelet,* Quantização e Codificação), os bits são transmitidos para o decodificador. Os dados transmitidos são decodificados, quantizados inversamente e é realizada a transformada *wavelet* inversa dos coeficientes quanizados, Fig. 2.1. Desta forma, a imagem é reconstruída.

Para se obter uma boa eficiência do processo é necessário que a imagem reconstruída seja uma boa aproximação da imagem original. Assim, algumas medidas são usadas para avaliar o conteúdo da informação contido na imagem reconstruída, bem como para estimar o desempenho dos sistemas de compressão envolvidos.

2.5 Medidas de Distorção em Compressão de Imagens

Para avaliar o desempenho dos sistemas de compressão de imagens estudados, são utilizadas duas medidas objetivas. A primeira é o Erro Quadrático Médio (MSE – *Mean Square Error*), que é calculado como

$$MSE = \frac{1}{VH}\sum_{i=0}^{V-1}\sum_{j=0}^{H-1}\left(x(i,j)-\hat{x}(i,j)\right)^2, \tag{2.62}$$

onde $x(i,j)$ é a amostra da imagem original e $\hat{x}(i,j)$ é a amostra reconstruída. As imagens original e reconstruída são ambas de dimensões VxH. A segunda medida objetiva é a relação sinal ruído de pico (PSNR – *Peak Signal to Noise Ratio*) dada em decibéis por

$$PSNR(dB) = 10.\log_{10} \frac{M^2}{MSE}, \tag{2.63}$$

onde M é o maior valor pico a pico que o sinal de entrada pode assumir. Tipicamente M = 255 para imagens originais representadas com 8 bits/pixel.

Capítulo 3

Técnicas de Compressão de Imagens Utilizando *Wavelets*

A transmissão e o armazenamento de imagens ainda é um dos principais problemas para o desenvolvimento de sistemas de comunicação de multimídia devido à largura de banda e a memória exigidas para estes processos. Conseqüentemente, muitas técnicas de compressão de imagens tem sido desenvolvidas nas últimas décadas. Embora técnicas de compressão sem perda (reversível) sejam preferíveis, as taxas de compressão alcançadas por estes métodos são relativamente baixas (da ordem de 2 ou 3). Assim, é necessário o uso de técnicas de compressão com perda (irreversível), que ocasionam alguma distorção na imagem reconstruída. A eficiência de um codificador pode ser definida pela qualidade da imagem obtida para uma dada taxa de bits. Esta eficiência é geralmente alcançada ao custo de uma maior complexidade computacional.

Alguns algoritmos de compressão de imagens realizam uma decomposição da imagem em diferentes subbandas e codificam os coeficientes de cada subbanda utilizando codificação diferencial (DPCM), técnicas de varredura juntamente com codificação de corridas de zeros (*run-length*) e codificação de entropia. Dentre estes trabalhos podem ser apontados alguns pioneiros como Antonini [1], Daubechies [11], Woods e O'Neil [30] e Ghraravi e Tabatabai [31]. Entretanto, estes estudos deixam de lado uma importante propriedade que é a correlação existente entre as subbandas. Esta observação levou ao desenvolvimento de uma forma de codificação especialmente projetada para a decomposição em multiresolução de transformadas implementadas por bancos de filtros. Ela é baseada numa estrutura de dados chamada "árvore de zeros" (*zerotree*), que é análoga à varredura zig-zag e ao símbolo de fim de bloco (EOB) usado na codificação via DCT (JPEG).

Um eficiente algoritmo de compressão usando árvores de zeros foi implementado por Shapiro [32]. O algoritmo resultante desta combinação utiliza uma estrutura chamada E*mbedded Zerotree Wavelet* (EZW). A codificação EZW explora a natureza de multiresolução da decomposição *wavelet* criando uma forma totalmente nova de codificar imagens. Uma das propriedades deste codificador, relativa à compressão de dados, é que ela tende a compactar a energia dos dados de entrada dentro de um número relativamente pequeno de coeficientes

wavelets (esta propriedade pode ser equivalente à redução da correlação entre os coeficientes *wavelets*). Outra característica do codificador que utiliza a EZW[❖] é que ele cria uma representação completamente embutida (*embedded*) dos bits, isto é, uma representação na qual uma imagem de alta resolução também contém todas as resoluções mais grosseiras da imagem original. Assim, o codificador pode interromper a codificação quando uma desejada taxa de compressão for atingida ou alguma outra condição for satisfeita. Do mesmo modo, o decodificador pode interromper a decodificação a uma determinada taxa desejada. Esta é uma característica muito interessante deste tipo de codificação e é importante em termos de largura de banda, espaço de armazenamento, complexidade computacional, tempo e custo. Este método de compressão produz excelentes resultados sem requerer conhecimento a priori da imagem ou tabelas de codificação previamente armazenadas como na quantização vetorial, além de não requerer nenhum treinamento dos codificadores e decodificadores, como no padrão JPEG (para otimizar as tabelas de quantização).

Uma desvantagem do codificador EZW é que seu desempenho é muito pobre quando são introduzidos erros de bits dentro dos dados codificados. Isto acontece porque a natureza embutida do codificador causa propagação de erros do ponto que eles são introduzidos até o final dos dados. Isto não é um problema para ambientes com baixas probabilidades de erro, mas em ambientes sem fio, onde a probabilidade de erro é alta, seu desempenho pode ser muito baixo. Mas algumas técnicas baseadas no algoritmo EZW têm sido estudadas a fim de melhorar os resultados obtidos em meios com alta probabilidade de erro de bit. Um exemplo é o algoritmo REZW [33] estudado por Creusere. Este algoritmo é baseado no algoritmo EZW e alcança excelentes resultados em canais de comunicação ruidosos. Posteriormente, este algoritmo vai ser tratado com mais detalhe.

Uma outra desvantagem do algoritmo EZW é que a sua estrutura não é muito flexível. Por exemplo, algumas aplicações requerem que a imagem seja decodificada seletivamente para aumentar a resolução somente em certas partes da imagem. Tal decodificação espacial seletiva requer modificações no algoritmo EZW original.

Uma nova técnica, tal como o algoritmo EBCOT (*Embedded Block Coding with Optimized truncation*) o qual é usado no padrão JPEG 2000 [34], trata de alguma dessas falhas do algoritmo EZW. Uma outra modificação do algoritmo EZW, é o algoritmo SPIHT (*Set Partitioning in Hierarchical Trees*) proposto por Said e Pearlman [35]. Os resultados

[❖] Por simplicidade, adotou-se as expressões Codificador EZW e Algoritmo EZW.

encontrados por este método são melhores que os obtidos pelo algoritmo EZW. Este método será visto mais detalhadamente no apêndice A deste trabalho.

3.1 Algoritmos de Compressão de Imagens

3.1.1 O algoritmo EZW

O algoritmo EZW implementado por Shapiro [32], se divide em 3 processos básicos: 1) uma Transformada *Wavelet* Discreta ou uma decomposição em subbandas hierárquica; 2) predição da ausência de informação significativa através das escalas pela exploração das similaridades próprias da imagem; 3) Compressão sem perda da seqüência resultante, que é obtida via codificação aritmética adaptativa [29].

A estrutura geral do algoritmo de compressão de imagem EZW é mostrada na Fig. 3.1.

Fig. 3.1: Diagrama para uma codificação de imagem sem perda, baseada na EZW.

No primeiro estágio os dados da imagem original são descorrelacionados pelos bancos de filtros da transformada *wavelet*, para produzirem subbandas de correlação reduzida. Idealmente, se a transformada *wavelet* removesse toda correlação entre as amostras, não seria necessário codificar as subbandas usando *zerotrees*. Os dados transformados seriam codificados simplesmente usando técnicas de codificação de entropia. Entretanto, na prática, depois da transformada *wavelet* ainda existe uma alta correlação dentro das subbandas e entre as subbandas. A correlação dos coeficientes entre as subbandas é mostrada na Fig. 3.2. As subbandas de detalhamento 2, 5 e 8 são bastante correlacionadas entre si, uma vez que a subbanda 2 é uma aproximação grosseira da subbanda 5, e esta por sua vez, é uma aproximação grosseira da subbanda 8. O mesmo é válido para as subbandas 3, 6 e 9 e para as subbandas 4, 7 e 10.

A correlação entre as subbandas é mostrada da seguinte forma, supondo que o pixel no canto superior esquerdo da subbanda 2 da Fig. 3.2 seja de valor reduzido, observa-se da figura que há uma grande probabilidade dos pixels do quadrado 2x2 da subbanda 5 também serem

pequenos. Da mesma forma, os pixels do quadrado 4x4 da subbanda 8 provavelmente também serão pequenos. Essas regiões quadradas são denominadas regiões espacialmente correlacionadas.

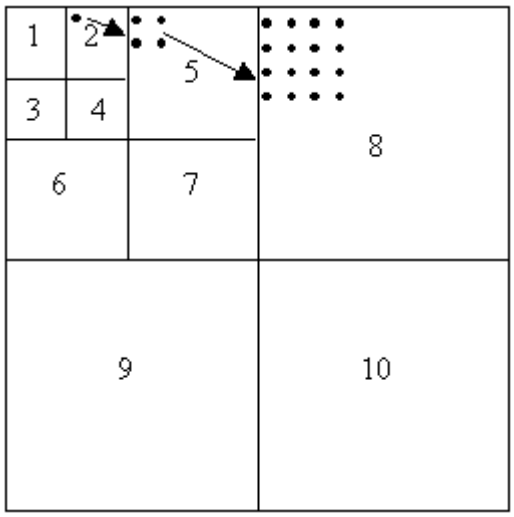


Fig. 3.2: Correlação dos coeficientes entre as subbandas
para 3 níveis de decomposição.

O segundo estágio da estrutura geral do algoritmo de compressão EZW explora a correlação existente entre as subbandas. O papel principal da codificação EZW é efetuar a quantização dos coeficientes das *wavelets* e reordená-los de forma que possam ser codificados com uma maior eficiência no estágio seguinte. Um codificador EZW deve sempre ser seguido por um codificador de entropia. O terceiro estágio, que consiste na codificação aritmética do algoritmo proposto por Shapiro [32], explora a correlação residual usando técnicas de modelagem adaptativas [29].

Todos os três estágios precisam trabalhar em conjunto para se alcançar uma melhor eficiência na compressão. No entanto, a codificação EZW tem o papel principal para se alcançar uma compressão eficiente.

3.1.1.1 Codificador EZW

O codificador EZW foi especialmente projetado para trabalhar juntamente com a transformada *wavelet*, o que explica a inclusão da palavra *wavelet* em seu nome. Ele foi

proposto para operar em imagens (sinais 2-D), mas isto não impede que também seja usado em sinais de outras dimensões.

Na aplicação em imagens, a transformada *wavelet* transforma um sinal do domínio espacial para um domínio conjunto espaço-escala. Isto significa que os coeficientes das *wavelets*[❖] unidimensionais (usando separabilidade) são bidimensionais, como foi visto no capítulo anterior. Para que o sinal transformado seja comprimido, não somente os valores dos coeficientes serão codificados, mas também suas posições espaciais.

O codificador EZW realiza uma codificação progressiva, comprimindo uma imagem dentro de um vetor de bits (*bitstream*), com um aumento progressivo de resolução. Isto significa que quanto mais bits são adicionados ao vetor, mais detalhes a imagem decodificada poderá conter. Codificação progressiva é também conhecida como codificação embutida (*embedded encoding*). O termo embutido é usado para indicar que os dados comprimidos são ordenados na ordem de importância visual.

3.1.1.2 A Estrutura do Codificador EZW

O codificador EZW é baseado nas seguintes observações : 1) imagens naturais, em geral, têm um espectro passa-baixa, ou seja, a energia se concentra nas baixas freqüências. Então os coeficientes *wavelets*, em média, serão menores nas subbandas de baixas freqüências que nas subbandas de altas freqüências. Isto faz com que a codificação progressiva através da transformada *wavelet* seja uma escolha natural para compressão de imagens, visto que as subbandas de frequências mais altas somente adicionam detalhes à imagem; 2) coeficientes *wavelets* maiores são mais importantes que coeficientes *wavelets* menores.

Estas duas observações são exploradas pelo codificador EZW, que codifica os coeficientes na ordem decrescente de magnitude. Os coeficientes da imagem são comparados com um limiar previamente estabelecido. Se o coeficiente for maior que o limiar, ele é codificado e removido da imagem. Se for menor, é deixado na imagem para ser codificado nas iterações futuras. O artifício é usar a dependência entre os coeficientes *wavelets* através das diferentes escalas para eficientemente codificar grandes partes da imagem que se encontram abaixo do limiar em questão. Este é o papel principal do codificador EZW. Após todos os coeficientes serem visitados (*scanned*) o limiar é reduzido e os coeficientes que estavam abaixo

❖ Por simplicidade adotou-se Coeficientes *Wavelets* no texto.

do limiar anterior são visitados novamente. Se agora forem maiores que o limiar atual, eles são codificados. Caso contrário, são deixados novamente na imagem para as próximas iterações. Este processo se repete até que todos os coeficientes sejam codificados ou outro critério seja satisfeito, como por exemplo, a máxima taxa de bits seja atingida.

No sistema de subbanda hierárquico, com exceção das subbandas de freqüências mais altas, cada coeficiente de uma dada escala pode ser relacionado a um conjunto de coeficientes da próxima escala mais fina de orientação similar. Os coeficientes localizados na escala mais grossa, (escala de baixas freqüências) são denominados "pais", e todos os coeficientes correspondentes à mesma localização espacial na próxima escala mais fina de orientação similar são chamados "filhos". Para um dado pai, o conjunto de todos os coeficientes em todas as escalas mais finas de orientação similar correspondendo à mesma localização espacial, são denominados descendentes. Para uma decomposição em subbadas QMF-piramidal, as dependências entre pai-filho são mostradas na Fig. 3.3.

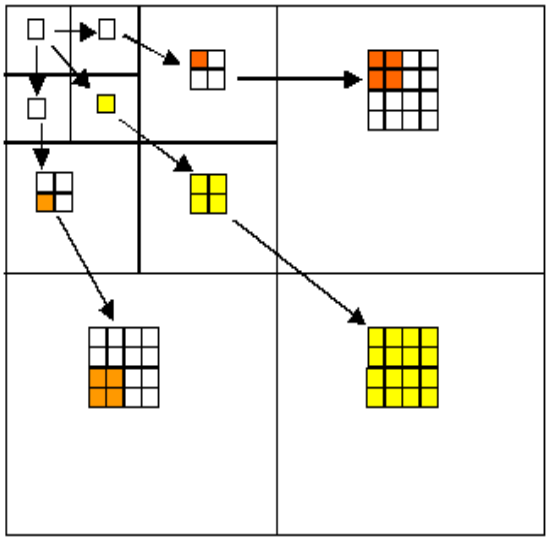


Fig. 3.3: Relação dos coeficientes em um codificador EZW.

Cada coeficiente pai tem 4 filhos (Fig. 3.3), com exceção da subbanda de baixas freqüências, onde cada pai tem 3 filhos, um em cada subbanda na mesma escala mas de diferente orientação.

Os coeficientes são visitados em uma ordem predeterminada de forma que nenhum coeficiente filho é visitado antes do seu pai, mas somente depois que seu pai e todos os pais vizinhos (localizados na mesma subbanda) forem visitados (Fig. 3.4). Para uma transformada de 3 escalas, a varredura começa na subbanda de freqüência mais baixa, denotada por LL_3, e

segue para as subbandas HL_3, LH_3, HH_3, como mostrado na Fig. 3.4. Deste ponto ela segue para a escala 2, e assim por diante. Nota-se que todos os coeficientes dentro de uma dada escala são visitados antes dos coeficientes da próxima escala.

Se um coeficiente *wavelet x* na escala mais grossa (subbanda de baixas freqüências) é insignificante com respeito a um limiar *T*, isto é, se $|x| < T$, então todos os coeficientes *wavelets* descendentes deste nó serão, possivelmente, insignificantes com respeito a *T*. Evidências empíricas sugerem que esta hipótese é verdadeira. Assim todos os coeficientes de mesma localização espacial, como mostrado na Fig. 3.3, serão codificados com um único símbolo denominado *Zerotree* (árvore de zeros). *Zerotree* é uma árvore na qual todos os seus nós representam coeficientes iguais ou menores que o limiar. Uma "raiz *zerotree"* é codificada com um símbolo especial indicando que a insignificância dos seus descendentes é completamente previsível.

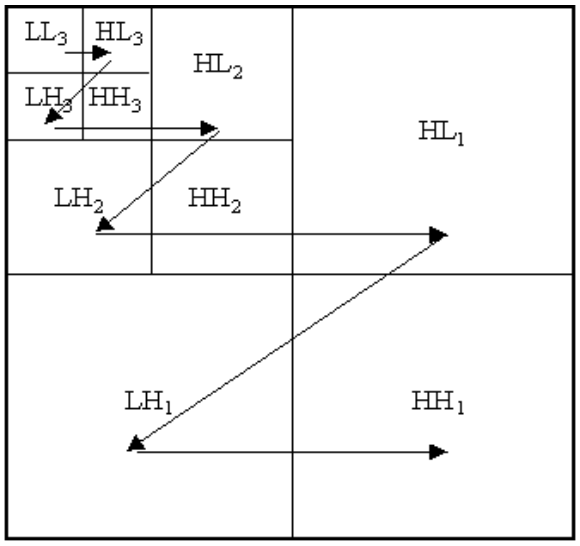


Fig. 3.4: Ordem de varredura (*scanning*) dos coeficientes *wavelets* nas subbandas.

O codificador EZW explora a *zerotree* com base na observação que os coeficientes tendem a decrescer com a escala. Considera-se que haverá maior probabilidade de que um coeficiente na árvore seja menor que um certo limiar se sua raiz for menor que este limiar. Se este for o caso, então toda a árvore pode ser codificada com um único símbolo *zerotree*, alcançando assim alta compressão. Se a imagem é visitada em uma ordem predefinida, indo da escala mais alta para a mais baixa (Fig.3.4), implicitamente muitas posições serão codificadas através do uso de *zerotrees*. Na prática, em muitos casos não se obtêm *zerotrees*, mas a probabilidade de que ela ocorra é geralmente alta.

Na codificação é realizada uma quantização, que está relacionada com a codificação das magnitudes dos coeficientes, chamada quantização de aproximação sucessiva (SAQ). O primeiro passo é determinar o limiar inicial e, aos poucos, a seqüência de limiares (T_0, ..., T_{N-1}) para que a significância dos coeficientes seja determinada. A seqüência de limiares usada é uma seqüência de potências de dois, chamada codificação *bitplane* [36]. Sendo assim, o limiar inicial T_0 é escolhido tal que

$$T_0 = 2^{\lfloor \log_2 x_{\max} \rfloor}, \tag{3.1}$$

onde x_{max} corresponde ao coeficiente de valor máximo na imagem, e $\lfloor . \rfloor$ significa o menor inteiro maior que o argumento. Os limiares das iterações seguintes são reduzidos consecutivamente por um fator de 2, i. e.,

$$T_i = T_{i-1} / 2 . \tag{3.2}$$

Durante a codificação (e decodificação) duas listas separadas de coeficientes *wavelets* são formadas, a lista dominante e a lista subordinada. A lista dominante contém as coordenadas daqueles coeficientes que ainda não foram considerados significantes com respeito ao limiar, ou seja, que ainda não foram codificados. Esta lista é formada na mesma ordem em que os coeficientes estão sendo visitados. A lista subordinada contém as magnitudes dos coeficientes que foram classificados como significantes, ou seja, aqueles valores encontrados superiores ao limiar em módulo. Para cada limiar, cada lista é incrementada uma vez durante o processo de codificação.

Na codificação (e decodificação) há também duas passagens: uma passagem dominante e uma passagem subordinada. A passagem dominante encontra os valores dos pixels que estão acima do limiar, enquanto a passagem subordinada quantiza todos os valores dos pixels significantes encontrados nas passagens dominantes anteriores.

Durante uma passagem dominante, os coeficientes com coordenadas na lista dominante, que ainda não foram tidos como significantes, são comparados com o limiar "T_i" para determinar suas significâncias. Se significantes, os sinais (positivo ou negativo) destes coeficientes são codificados. Valores de pixels tidos como significantes na passagem dominante são codificados com o símbolo positivo para um valor maior que zero, ou um símbolo negativo, para um valor menor que zero. Estes pixels são então adicionados à lista subordinada para

serem quantizados, e as posições deles na imagem são preenchidas com zeros, evitando assim, que sejam codificados novamente na próxima passagem dominante. Valores de pixels encontrados sendo insignificantes na passagem dominante, mas com filhos significantes, são codificados como Zeros Isolados, e seus descendentes são codificados individualmente. Quando o valor de um pixel é visto como insignificante, e todos os seus descendentes também o são, é possível codificar aquele pixel e toda sua geração com um único símbolo *zerotree*, como visto anteriormente. Assim, a passagem dominante mapeia os valores dos pixels em 4 símbolos do alfabeto os quais podem ser codificados com o uso de um codificador aritmético adaptativo [29]. Os quatro símbolos usados são:

1) POS (Positivo Significante): o coeficiente é maior ou igual ao limiar atual, com sinal positivo, e não foi codificado nas iterações anteriores;
2) NEG (Negativo Significante): o coeficiente é maior ou igual ao limiar atual, com sinal negativo, e não foi codificado anteriormente;
3) IZ (Zero Isolado): o coeficiente é menor que o limiar atual (i. e., é insignificante), mas um ou mais dos seus descendentes não o são;
4) ZTR (Raiz *Zerotree*): o coeficiente (que no caso é a raiz da árvore) e todos os seus descendentes são insignificantes com respeito ao limiar, ou somente ele é insignificante, se este estiver localizado em uma das subbandas de mais alta freqüência (LH_1, HL_1 ou HH_1, Fig. 3.4).

Cada passagem dominante é seguida por uma passagem subordinada, que refina a magnitude de todos os coeficientes que se encontram na lista subordinada, alcançando assim uma maior precisão. Durante uma passagem subordinada, a largura do degrau do quantizador, que define o intervalo de incerteza para a magnitude real do coeficiente, é dividida pela metade. Para cada magnitude encontrada na lista subordinada, este refinamento pode ser codificado usando um alfabeto binário com o símbolo “1” indicando que o valor verdadeiro do coeficiente está na metade superior do intervalo de incerteza ou um símbolo “0” indicando que o valor verdadeiro deste coeficiente se encontra na metade inferior.

O processo continua alternando entre uma passagem dominante e uma passagem subordinada, enquanto o limiar vai sendo reduzido pela metade. A codificação pára quando

uma determinada condição é satisfeita, tal como quando a taxa de bits planejada ou a distorção máxima consentida é encontrada.

3.1.1.3 Exemplo do Algoritmo EZW

Esta seção apresenta os detalhes do algoritmo EZW através de um exemplo simples. Um outro exemplo bastante didático pode ser encontrado em [37]. Os coeficientes a serem codificados são de uma transformada *wavelet* de dois níveis, de uma imagem de tamanho 4x4, como mostrado na Fig. 3.5.

17	6	8	13
3	-5	-7	1
2	2	-6	4
1	-2	3	-2

Fig. 3.5: Coeficientes *wavelets* de uma imagem 4x4.

O coeficiente máximo da Fig. 3.5 é 17, o qual resulta em um limiar inicial de $T_0 = 16$ (Eq. 3.1).

Os resultados da primeira passagem dominante são mostrados na Tab. 3.1.

Passagem Dominante 1 (T_0=16)				
Subbanda	Valor de Coeficiente	Símbolo	Valor de Reconstrução	Comentários
LL_2	17	POS	24	1)
HL_2	6	ZTR	0	2)
LH_2	3	ZTR	0	2)
HH_2	-5	ZTR	0	2)

Tab. 3.1: Resultados da primeira passagem dominante.

Os comentários a seguir referem-se às linhas de comentários indicadas na Tab. 3.1.

1) O coeficiente tem uma magnitude maior que o limiar T_0 e é positivo. Assim, o símbolo resultante é positivo significante (POS), e o decodificador reconhece que este símbolo está no intervalo [T_0 , $2T_0$), ou seja, [16,32). O valor de reconstrução para este coeficiente se encontra na metade do intervalo, isto é, 24.
2) O coeficiente e todos os seus descendentes são menores que o limiar T_0. Assim, estes coeficientes são codificados como uma raiz *zerotree* (ZTR). Com isto, o coeficiente e seus descendentes são temporariamente considerados como zero.

Na primeira passagem subordinada o codificador envia os bits 0 ou 1 para indicar se o coeficiente significante (neste exemplo, 17) está no intervalo inferior [16,24) ou no intervalo superior [24,32). Neste exemplo, o símbolo enviado é 0 indicando que o coeficiente se encontra na metade inferior do intervalo de incerteza. O resultado da primeira passagem subordinada é mostrado na Tab. 3.2.

Passagem Subordinada 1		
Magnitude do Coeficiente	Símbolo	Magnitude de Reconstrução
18	0	20

Tab. 3.2: Resultados da primeira passagem subordinada.

Para a segunda passagem dominante o coeficiente 17 não precisa ser codificado novamente. Assim, a transformada *wavelet* aparece como na Fig. 3.6, onde o coeficiente 17 é substituído por um 0 para que ele não seja codificado novamente.

0	6	8	13
3	-5	-7	1
2	2	-6	4
1	-2	3	-2

Fig. 3.6: Imagem 4x4 após a primeira passagem dominante.

O limiar para a segunda passagem dominante é T_1=8 (ou seja, T_0/2) e os resultados desta passagem estão listados na Tab. 3.3.

Passagem Dominante 2 (T_1=8)				
Subbanda	Valor de Coeficiente	Símbolo	Valor de Reconstrução	Cometários
HL_2	6	IZ	0	3)
LH_2	3	ZTR	0	
HH_2	-5	ZTR	0	
HL_1	8	POS	12	4)
HL_1	13	POS	12	4)
HL_1	-7	ZTR	0	4)
HL_1	1	ZTR	0	4)

Tab. 3.3: Resultados da segunda passagem dominante.

Os seguintes comentários se aplicam:

3) O coeficiente é menor que o limiar mas dois dos seus descentes são maiores que o limiar. Assim o símbolo resultante é o zero isolado (IZ).
4) Estes coeficientes são descendentes do coeficiente que foi codificado como um zero isolado.

Na segunda passagem subordinada o coeficiente quantizado na passagem subordinada anterior (17) é refinado, se aproximando assim do seu valor verdadeiro. Como ele estava na metade inferior do intervalo de incerteza anterior agora o novo intervalo para este coeficiente é [16,24).

Os coeficientes a serem quantizados nesta passagem são 8 e 13 os quais são maiores que o limiar atual T_1. Estes coeficientes se encontram no intervalo de incerteza [8,16). Os resultados da segunda passagem dominante são mostrados na Tab. 3.4.

Passagem Subordinada 2		
Magnitude do Coeficiente	Símbolo	Magnitude de Reconstrução
18	0	18
8	0	10
13	1	14

Tab. 3.4: Resultados da segunda passagem subordinada.

Uma nova passagem dominante é iniciada, e o processo continua alternando entre uma passagem dominante e uma passagem subordinada até que toda a imagem seja codificada ou alguma condição previamente estabelecida (máxima taxa de bits ou limiar de distorção aceitável) seja encontrada.

3.1.1.4 A Decodificação

As informações mínimas exigidas pelo decodificador são o número de decomposições que foram realizadas na transformada *wavelet*, as dimensões da imagem e o limiar inicial. Adicionalmente, pode também ser informada a média da imagem. Esta informação é útil se a média for subtraída da imagem antes da codificação. Depois da reconstrução do sinal de média nula o decodificador pode então somar a média original à imagem reconstruída.

Não é necessário enviar ao decodificador as posições dos coeficientes, pois com as informações iniciais (número de decomposições *wavelet* e dimensões da imagem,) o decodificador pode determinar as posições dos coeficientes.

Com estas informações o decodificador primeiramente gera uma matriz de zeros com as mesmas dimensões da imagem original. Aos poucos esta matriz de zeros é substituída pelos coeficientes que estão sendo quantizados inversamente. Cada símbolo decodificado, ambos durante uma passagem subordinada e uma passagem dominante, refina e reduz a largura do intervalo de incerteza no qual o valor verdadeiro do coeficiente (ou coeficientes, no caso de uma raiz *zerotree*) deve ocorrer. O valor reconstruído pode estar em qualquer lugar no intervalo de incerteza. Para que se tenha uma distorção mínima, usa-se o centro do intervalo de incerteza como o valor reconstruído. A decodificação pára quando todo o vetor de bits for enviado ou uma taxa de bits máxima for alcançada.

3.1.2 Método Robusto de Compressão de Imagens Baseado no Algoritmo EZW

Nesta seção descreve-se um método robusto de compressão de imagens baseado no algoritmo EZW. Como visto anteriormente, o algoritmo EZW alcança bons resultados em canais de baixa probabilidade de erro, mas em canais de comunicação sem fio onde a probabilidade de erro é alta seu desempenho é baixo. Assim, um novo método de compressão de imagens baseado no algoritmo EZW, denominado REZW (*Robust Embedded Zerotree*

Wavelet), foi investigado com a intenção de melhorar o desempenho em canais de comunicação ruidoso. A descrição segue de perto a contribuição original de Creusere [33]. Sua robustez é alcançada pela divisão dos coeficientes transformados em grupos que são independentemente processados usando um codificador embutido (*embedded*). Assim, quando ocorre um erro de bit em um dos grupos, os bits de outros grupos não são afetados, permitindo que mais informação correta chegue ao decodificador.

Este algoritmo foi especialmente implementado para ser usado em canais de comunicação sem fio, onde há uma maior necessidade de proteção a erros durante a transmissão. Recentemente, a proliferação de serviços sem fio e da Internet juntamente com a demanda por produtos de multimídia tem estimulado o interesse na transmissão de imagens e de vídeo em canais ruidosos de comunicação cuja capacidade varia com o tempo. Em tais aplicações, é vantajoso combinar o processo de codificação da fonte e do canal (i. e., compressão de dados e correção de erros).

A base do algoritmo REZW foi tirada do algoritmo EZW original, mas ele pode ser aplicado a outros codificadores baseados em *wavelets*, tal como o algoritmo SPIHT [35].

O codificador REZW é vantajoso em relação ao algoritmo EZW porque no decodificador EZW todos os bits recebidos antes de ocorrer o primeiro erro são usados na reconstrução da imagem, mas todos que são transmitidos depois são descartados. No algoritmo REZW, no entanto, os coeficientes *wavelets* são codificados em múltiplas e independentes seqüências de bits. Assim um único erro de transmissão trunca apenas uma das seqüências e as outras seqüências podem ainda ser completamente recuperadas pelo decodificador. Conseqüentemente, os coeficientes *wavelets* representados pelas seqüências truncadas são reconstruídos em resolução reduzida, enquanto aqueles que correspondem às seqüências que foram completamente recebidas pelo decodificador, ou seja, os coeficientes representados pelas seqüências que não possuem erros de transmissão, são reconstruídos com sua resolução total. Como os conjuntos de coeficientes em cada seqüência se estendem pela imagem inteira, então a transformada *wavelet* inversa igualmente mistura as diferentes resoluções resultando em uma boa qualidade da imagem.

O codificador REZW divide os coeficientes *wavelets* em S grupos (vetores de bits), quantizando-os e codificando-os independentemente [33]. Estes vetores de bits são então intercalados (*interleaved*), como for apropriado (por exemplo, em bits, bytes, pacotes, etc.), antes da transmissão, de forma que a natureza embutida do vetor de bits seja mantida. O

intercalador é um sistema de reordenamento dos dados que busca manter a característica de codificação embutida do codificador original. Para que o algoritmo REZW seja eficaz, cada grupo de coeficientes *wavelets* deve ter tamanho igual e deve codificar uniformemente a imagem. Deste modo, a ocorrência de um erro e conseqüente perda de um dos grupos, fará com que a distorção se distribua igualmente sobre a imagem.

A Fig.3.7 mostra a estrutura básica do algoritmo REZW.

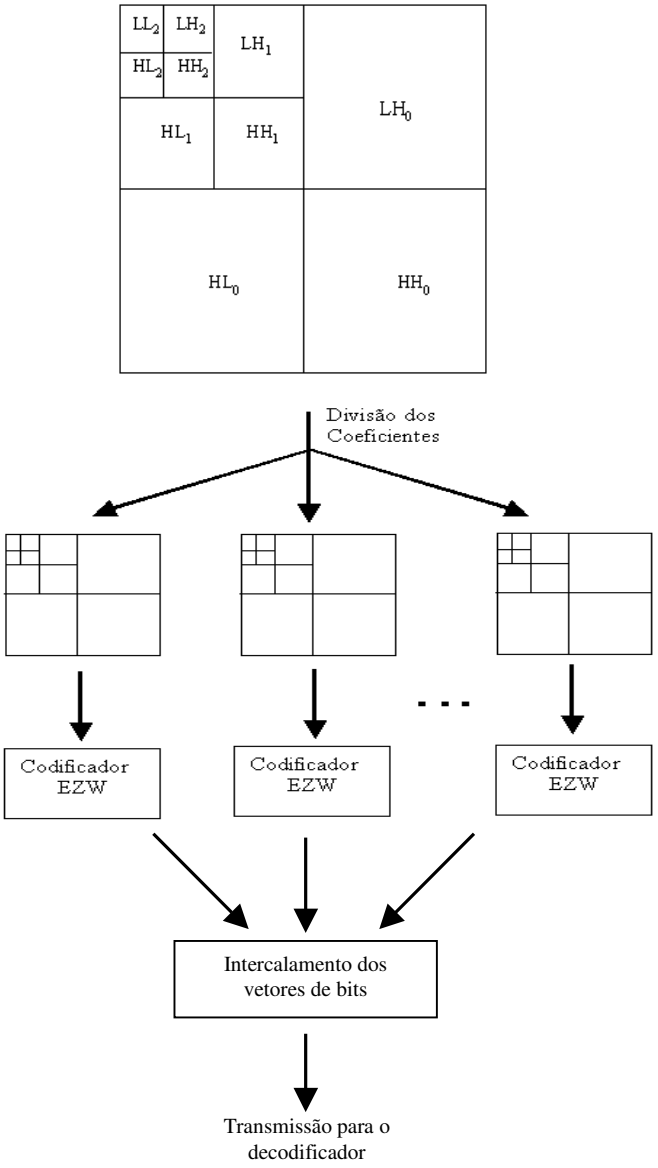


Fig. 3.7: Estrutura do algoritmo REZW.

No algoritmo REZW, a divisão dos coeficientes *wavelets* em grupos pode ser feita de duas maneiras: 1) *Zerotree Preserving* – ZP, onde a relação de descendência do algoritmo EZW

é preservada; 2) *Offset Zerotree* – OZ, onde a relação de descendência da EZW não é mantida [33].

***Zerotree Preserving – ZP*:** A Fig. 3.8 ilustra graficamente a divisão dos coeficientes *wavelets* em 4 grupos (S = 4) para uma transformada *wavelet* de 3 escalas. Na Fig. 3.8, cada coeficiente da mesma cor mapeia o mesmo grupo e é portanto, processado pelo mesmo codificador. Ademais, os quadrados marcados com um x são usados para definir os elementos de uma *zerotree*.

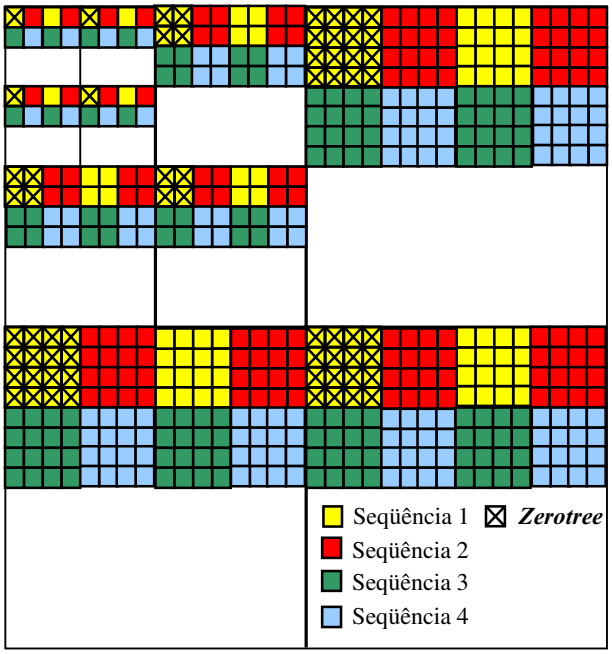


Fig. 3.8: Estrutura ZP para S = 4 em uma decomposição em 3 escalas *wavelets*.

Como pode-se observar, todos os elementos de uma dada *zerotree* são introduzidos no mesmo codificador e assim, a correlação entre escalas dos coeficientes insignificantes é totalmente explorada. Nota-se que S pode ser incrementado por potências de 4 até que o codificador processe cada vetor com somente uma *zerotree*. Se a imagem tem tamanho "XxY" (supondo, por simplicidade, que X e Y são potências de 2) e M escalas de decomposições *wavelets* são usadas, então o número máximo de vetores independentes permitidos é $S = X.Y / 4^M$.

Introduzir mais seqüências de bits é vantajoso em termos de robustez a erros. Entretanto, o desempenho taxa × distorção é prejudicado quando a probabilidade de erro de transmissão é

baixa. A divisão das escalas j dos coeficientes *wavelets* que mapeiam $W_j(x,y)$ dentro de $S = 4^k$ grupos, onde k é um inteiro, é dado por

$$W_{\Psi,j}(x,y) = W_j \begin{pmatrix} 2^{M-j-1}\left\{\left\lfloor \frac{x}{2^{M-j-1}} \right\rfloor .(2^k - 1) + n\right\} + x, \\ 2^{M-j-1}\left\{\left\lfloor \frac{y}{2^{M-j-1}} \right\rfloor .(2^K - 1) + m\right\} + y \end{pmatrix} \tag{3.3}$$

onde $\Psi = n + 2^k m$ específica o número da seqüência para $\{n,m\} \in [0,2^k - 1]$ e $\lfloor . \rfloor$ denota o maior inteiro menor que o argumento [33]. A escala j é inversamente proporcional à freqüência, isto é, j = 0 e j = M-1 se referem às subbandas de freqüências mais alta e mais baixa, respectivamente.

***Offset Zerotree* – OZ :** Neste caso, a divisão dos coeficientes *wavelets* em quatro grupos é ilustrada na Fig. 3.9.

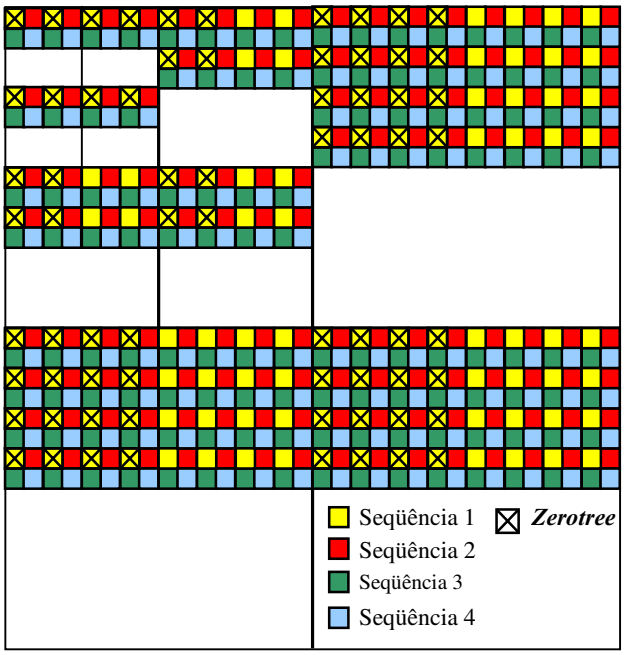


Fig. 3.9 : Estrutura OZ para S = 4 em uma decomposição em 3 escalas *wavelets*.

Nesta divisão, a relação de descendência da estrutura *zerotree* convencional não é preservada. Em vez disso, a estrutura de descendência se distribui entre as escalas *wavelets* de tal forma que nenhum coeficiente adjacente é colocado no mesmo grupo. No caso mais geral onde $S = 4^k \leq X.Y / 4^M$, para k inteiro, cada coeficiente em um dado grupo está separado de outro coeficiente do mesmo grupo por no mínimo $2^k - 1$ coeficientes *wavelets*. Na distribuição ZP, cada 2^k ésima amostra (em ambas as direções horizontal e vertical) das quatro escalas mais grosseiras são mapeadas no mesmo grupo. Na distribuição OZ, entretanto, a reamostragem que ocorre nas quatro escalas mais grosseiras é mantida em todos as outras escalas *wavelets*, i.e., a equação (3.3) é aplicada em todas escalas com $j = M - 1$.

A vantagem deste método em relação ao ZP é que se um vetor de bits é terminado prematuramente por erro de transmissão, os coeficientes de resolução reduzida correspondentes estarão rodeados pelos coeficientes de resolução completa, que foram transmitidos em seqüências independentes. Assim, a imagem reconstruída pela transformada inversa conterá mais informação de alta freqüência quando um erro ocorrer durante a transmissão. Entretanto, as *zerotrees* originais ficam espacialmente alternadas entre escalas, reduzindo a correlação entre escalas dos coeficientes *wavelets* e, conseqüentemente, o desempenho taxa × distorção do algoritmo de codificação.

3.2 Avaliação Probabilística

Para avaliar a efetividade do algoritmo de compressão REZW, considera-se que a imagem codificada é transmitida através de um canal binário simétrico sem memória com probabilidade de erro ε. Deseja-se saber o número de bits que foram corretamente recebidos em cada um dos S grupos de vetores. Visto que estes números são variáveis aleatórias, utilizam-se os valores médios para caracterizar o desempenho dos diferentes algoritmos. Devido ao fato do canal ser sem memória as seqüências são afetadas de modo independente umas das outras. Considerando-se que a imagem é comprimida para um total de B bits e que S seqüências são usadas, então a probabilidade de receber corretamente k dos B/S bits de cada vetor é dada por [33]

$$p(k) = \begin{cases} \varepsilon.(1-\varepsilon)^k, & 0 \le k < \dfrac{B}{S} \\ (1-\varepsilon)^k, & k = \dfrac{B}{S} \end{cases}. \qquad (3.4)$$

Esta é uma distribuição de massa de probabilidade válida, como pode ser facilmente verificado pela soma dos termos com k variando de 0 a B/S. Na Eq. 3.4, $(1-\varepsilon)^k$ é a probabilidade de que os primeiros k bits transmitidos sejam recebidos corretamente, enquanto ε é a probabilidade que o $(k+1)$-ésimo bit esteja errado. O termo condicionado em k=*B*/*S* corresponde ao caso em que todos os bits da seqüência sejam corretamente recebidos. O valor médio pode ser calculado como

$$m_S = \sum_{k=0}^{B/S} k.p(k). \qquad (3.5)$$

Em média, o número total de bits corretamente recebidos é $S.m_S$. Se *B*/*S* é relativamente grande em relação a $1/\varepsilon$, então $m_S \approx m_1$ para qualquer valor de *S*, e portanto, aproximadamente *S* vezes mais bits são corretamente decodificados no algoritmo robusto que no algoritmo convencional (i.e., *S* = 1).

Nesta sessão apresentou-se um algoritmo de compressão de imagens em que uma certa robustez a erros de transmissão pode ser incorporada com um pequeno custo de complexidade. À medida que o número de seqüências independentes aumenta, a robustez a erros de transmissão também aumenta, mas o desempenho taxa × distorção do codificador-decodificador decresce quando não há erros de transmissão. Se a taxa de erro de bit é alta, entretanto, o desempenho deste algoritmo é geralmente melhor que o do algoritmo EZW convencional para valores de S maiores que um, como mostrado em [33].

Capítulo 4

Algoritmo Modificado - Simulações e Resultados

Neste capítulo propõe-se uma variação do algoritmo de codificação de imagens baseado no algoritmo EZW [32] apresentado no capítulo anterior. O algoritmo proposto é uma versão modificada do algoritmo EZW Robusto (REZW), doravante denominado algoritmo modificado, que utiliza um codificador de Huffman em vez do codificador aritmético adaptativo utilizado no algoritmo REZW [33]. O codificador de Huffman foi escolhido pela sua simplicidade e eficiência. Outro fator que influenciou a escolha é que o codificador aritmético é protegido por patentes, enquanto o codificador de Huffman é de domínio público.

Nas próximas sessões apresentam-se o algoritmo e os resultados das simulações obtidas com o mesmo. Estes resultados são apresentados para todos os possíveis valores de S (número de seqüências) e diferentes valores de probabilidade de erros de bit de transmissão. Também é realizada uma comparação dos resultados obtidos por este algoritmo para uma seqüência única (S=1) com os resultados apresentados por Shapiro (algoritmo EZW), para várias taxas de bit.

4.1 Algoritmo Modificado

No algoritmo modificado o limiar de codificação inicial é definido pela Eq. 3.1. A partir do limiar inicial e dos limiares subseqüentes, obtidos através da Eq. 3.2, os coeficientes são classificados com um dos símbolos do alfabeto {POS, NEG, IZ e ZTR} em todas as subbandas, com exceção da última subbanda onde ele é representado com 3 símbolos apenas. Como a última subbanda não possui descendentes, os símbolos IZ (Zero Isolado) e ZTR (*Zerotree*) são reunidos para formar um único símbolo denominado Z (Zero). Assim o alfabeto do codificador para a última subbanda é formado pelos símbolos {POS, NEG e Z}, permitindo maior compressão. Pode-se perceber que dois alfabetos são então utilizados na codificação da imagem.

Adiciona-se aos alfabetos um novo símbolo, denominado símbolo de parada. Este símbolo nunca é transmitido pelo codificador e é usado apenas para identificar a ocorrência de

erros de bits. Assim, quando ocorre um erro durante a transmissão a seqüência se torna aleatória de modo que o símbolo de parada é encontrado. Tudo que surge na recepção depois deste símbolo é descartado. Observa-se que, dependendo da posição do erro de bit, o decodificador de Huffman pode não encontrar o símbolo de parada. Neste caso, o decodificador continua o processo de decodificação com um sincronismo falso, e produz continuamente erros na reprodução. Para evitar esta situação, pode se usar um embaralhador (*scrambler*) [38] adicionado ao codificador antes da transmissão. No decodificador, o processo inverso é realizado no sinal de chegada através do desembaralhador (*descrambler*). Quando ocorre um erro de bit o desembaralhador não consegue recuperar seu sincronismo e passa a produzir bits aleatórios, fazendo com que o símbolo de parada seja encontrado com alta probabilidade.

Experimentos empíricos foram realizados a fim de escolher um código de Huffman ótimo. Após vários experimentos escolheu-se os seguintes mapas de significâncias:

1) Para todas as subbandas com exceção da última:
   - 1 – ZTR
   - 01 – IZ
   - 001 – POS
   - 0001 - NEG
   - 0000 - Parada

2) Para a última subbanda:
   - 1 – Z
   - 01 – POS
   - 001 – NEG
   - 000 – Parada

O comprimento do símbolo de parada foi escolhido de forma a não prejudicar o comprimento dos outros símbolos do código e de forma a garantir que ele seja encontrado quando ocorrer um erro de transmissão.

Os símbolos de cada seqüência são então codificados pelo codificador de Huffman e os bits resultantes da codificação são entrelaçados. Este entrelaçamento é realizado a fim de manter a natureza embutida do codificador, i.e., sua resolução progressiva.

As seqüências de bits codificadas são reagrupadas em uma única seqüência que é então transmitida através de um único canal ruidoso.

O vetor de bits é degradado pela soma de variáveis de ruído i.i.d. (independentes e identicamente distribuídas), segundo uma distribuição Bernoulli (ε) para diferentes valores do

parâmetro ε. O decodificador recebe o vetor de bits, separa-o no número de seqüências que foram geradas no codificador e faz o desentrelaçamento. Cada seqüência é decodificada separadamente até que o símbolo de parada seja encontrado ou até que a subseqüência chegue ao final. Se houver ocorrido erro de bit na transmissão, o decodificador encontrará o símbolo de parada, e os símbolos recebidos a partir deste ponto são descartados (não contribuem para a reconstrução da imagem). Caso contrário, toda a seqüência é decodificada e usada na reconstrução da imagem.

As seqüências de bits também são transmitidas separadamente em vários canais com diferentes probabilidades de erro, denominado método de transmissão multicanal.

A quantização utilizada foi a quantização uniforme como normalmente se faz com imagens. Observa-se que a quantização uniforme seguida de codificação de entropia tende a apresentar melhores resultados que a quantização de Lloyd-Max seguida de codificação de entropia [39].

Além das seqüências de bits, algumas informações adicionais devem ser enviadas ao decodificador para que a imagem original seja reconstruída. Estas informações são:

i) O número de decomposições *wavelets*;
ii) As dimensões da imagem;
iii) O limiar inicial (T).

Com estas informações, o decodificador primeiramente gera uma matriz de zeros com as mesmas dimensões da imagem original. Aos poucos esta matriz de zeros é substituída pelos valores parciais dos coeficientes que estão sendo reconstruídos inversamente. A cada passo a largura do intervalo de incerteza dos coeficientes é reduzida e refinada, se aproximando do valor verdadeiro do coeficiente da transformada *wavelet* da imagem original. Este processo continua até que o vetor de bits seja inteiramente decodificado ou uma taxa de bits máxima, previamente fixada, seja alcançada.

Após este processo, aplica-se a transformada *wavelet* inversa no sinal recuperado pelo decodificador para que a imagem reconstruída seja obtida.

4.2 Resultados das Simulações Utilizando o Algoritmo Modificado

Nesta sessão apresentam-se os resultados objetivos e subjetivos das simulações realizadas com o algoritmo modificado proposto neste trabalho.

As imagens de teste utilizadas foram as imagens Lena e Bárbara, ambas com dimensões $V = 512$ e $H = 512$, onde V é o número de pixels por coluna e H é o número de pixels por linha. Estas duas imagens encontram-se na Fig. 4.1 e em maior detalhe nas Figs. B.1 e B.2 que se encontram no Apêndice deste trabalho.

(a) Lena

(b) Bárbara

Fig 4.1: Imagens de teste.

Para as simulações utilizou-se a transformada *wavelet* biortogonal 9/7 (p=9 e q=7) [11] com 5 níveis de decomposição. Os coeficientes da *wavelet* biortogonal 9/7 são mostrados na Tab. 4.1. Somente são mostrados o coeficiente central e os coeficientes da direita, visto que o filtro é simétrico.

	***Wavelet* Biortogonal 9/7**
$\overline{g}$	0.6029; 0.2669; -0.0782; -0.0169; 0.0267
g	0.5575; 0.2956; -0.0288; -0.0456

Tab. 4.1: Coeficientes do filtro biortogonal 9/7.

Este filtro foi escolhido por apresentar maior eficiência na codificação de imagens sob o ponto de vista de distorção × taxa, segundo os estudos apresentados por Villasenor [40]. Por apresentar muito bom desempenho em compressão, este filtro tem sido usado extensivamente em aplicações de compressão de imagens [1], [40].

Foram consideradas todas as possíveis subdivisões do vetor de bits para a transmissão do sinal. Todas estas condições foram testadas para a taxa de 1 bit/pixel para as duas formas de divisões das seqüências: ZP (*Zerotree Partitioning*) e OZ (*Offset Zerotree Partitioning*), como visto no Capítulo 3.

As figuras a seguir mostram os resultados das simulações obtidos segundo critérios objetivos. Para cada uma destas condições apresentam-se a relação sinal/ruído de pico (PSNR) em função da probabilidade de erro de bit e exemplos das imagens reconstruídas correspondentes. Nas figuras são apresentados os gráficos obtidos com a codificação da imagem com o algoritmo modificado, e para efeito de comparação, também os resultados do algoritmo REZW. Estes resultados foram retirados dos gráficos apresentados em Creusere [33].

As avaliações dos resultados neste trabalho foram feitas com base na medida de PSNR (Relação Sinal de Pico/Ruído) que é uma medida de qualidade objetiva da imagem reconstruída. Outra possibilidade seria a avaliação baseada na qualidade subjetiva através de testes de opiniões de observadores. Neste caso a quantização dos coeficientes *wavelets* deve ser ajustada para que os intervalos de quantização aplicados aos vários coeficientes sejam compatíveis com a sensibilidade do sistema visual humano a estes coeficientes. Isto pode ser conseguido na codificação *zerotree* através de uma ponderação prévia dos coeficientes (antes de sua quantização) e da correção correspondente no decodificador.

As reproduções obtidas com S=1 são usadas como referências nas Figs. 4.4, 4.6, 4.8 e 4.10 para comparações com as reproduções obtidas com diferentes valores de S.

A Fig. 4.2 apresenta os resultados do algoritmo modificado e algoritmo REZW para uma seqüência apenas (S=1), codificada a uma taxa de 1 bit/pixel usando a divisão ZP (*Zerotree Partitioning*).

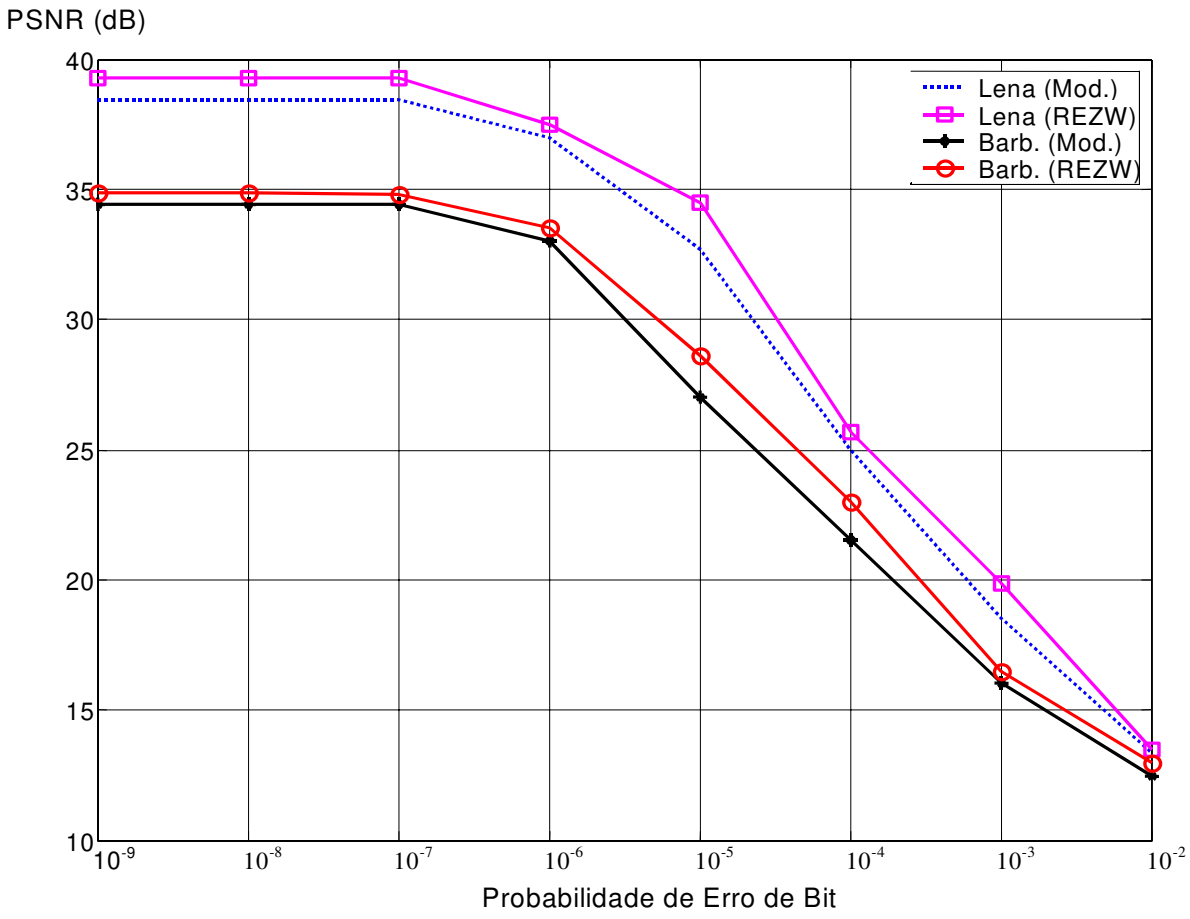


Fig. 4.2: PSNR × Probabilidade de Erro de Bit com S=1, codificada a uma taxa de 1bit/pixel usando divisão ZP.

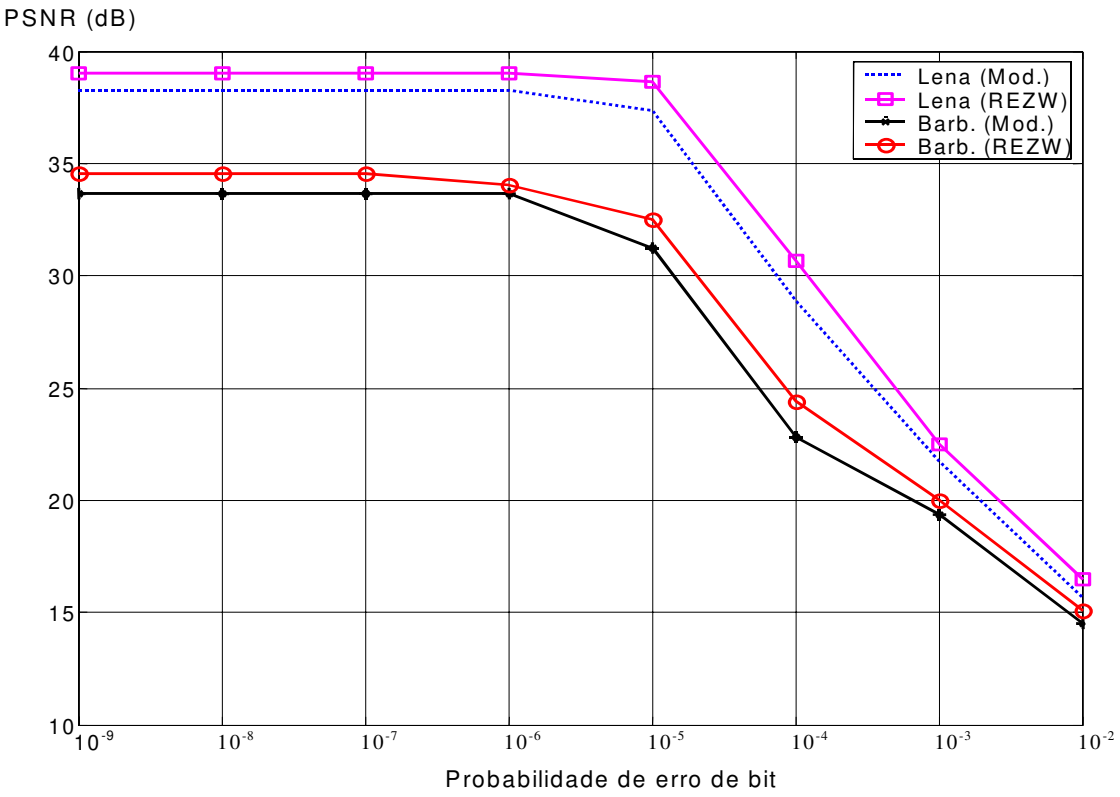


Fig.4.3: PSNR × Probabilidade de Erro de Bit para S=4, codificada a uma taxa de 1bit/pixel, usando a divisão ZP.

(a) (b)

Fig.4.4: Imagem Lena codificada com o algoritmo modificado a uma taxa de 1 bit/pixel usando divisão ZP.
(a) S=4, Probabilidade de erro de bit de 10^{-4}, PSNR = 28,89 dB;
(b) S=1, Probabilidade de erro de bit de 10^{-4}, PSNR = 24,95 dB.

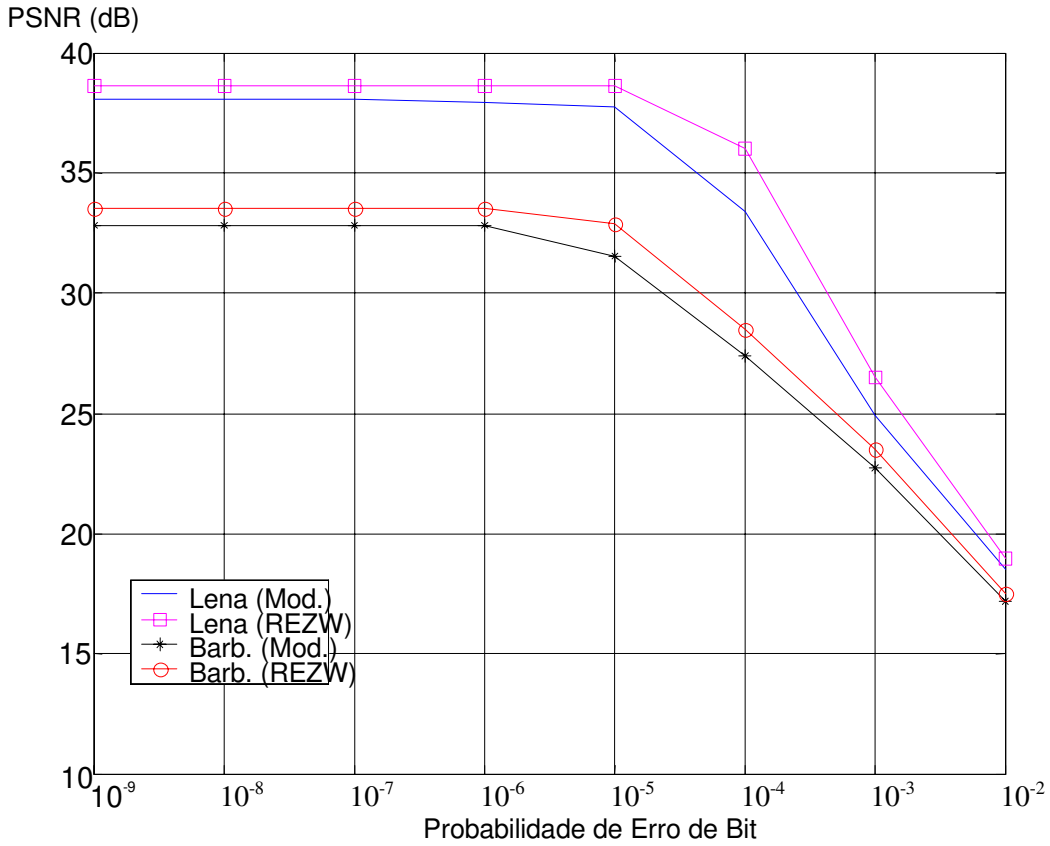


Fig.4.5: PSNR × Probabilidade de Erro de Bit com S=16, codificada a uma taxa de 1bit/pixel usando divisão ZP.

(a) (b)

Fig.4.6: Imagem Lena codificada com o algoritmo modificado a uma taxa de 1 bit/pixel usando divisão ZP.
(a) S=16, Probabilidade de erro de bit de 10^{-3}, PSNR = 24,93 dB;
(b) S=1, Probabilidade de erro de bit de 10^{-3}, PSNR = 18,53 dB.

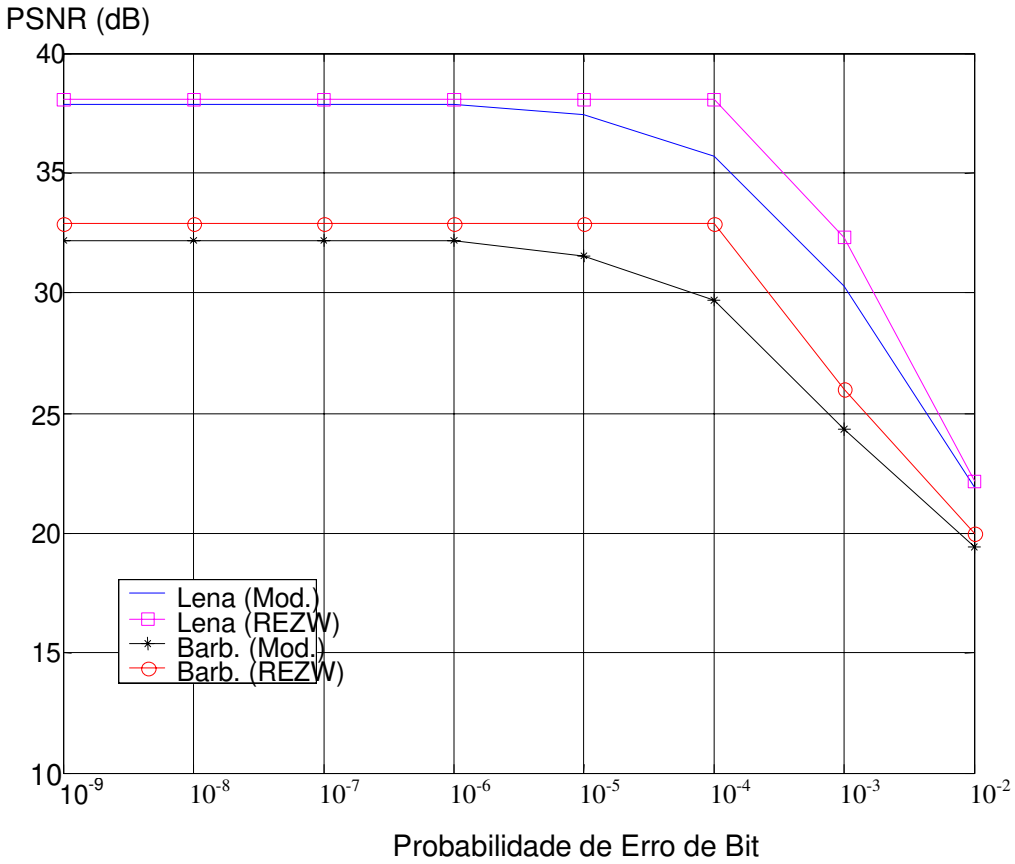


Fig.4.7: PSNR × Probabilidade de Erro de Bit com S=64, codificada a uma taxa de 1bit/pixel usando divisão ZP.

(a) (b)

Fig.4.8: Imagem Bárbara codificada com o algoritmo modificado a uma taxa de 1 bit/pixel usando divisão ZP.
(a) S=64, Probabilidade de erro de bit de 10^{-3}, PSNR = 24,32 dB;
(b) S=1, Probabilidade de erro de bit de 10^{-3}, PSNR=16,01 dB.

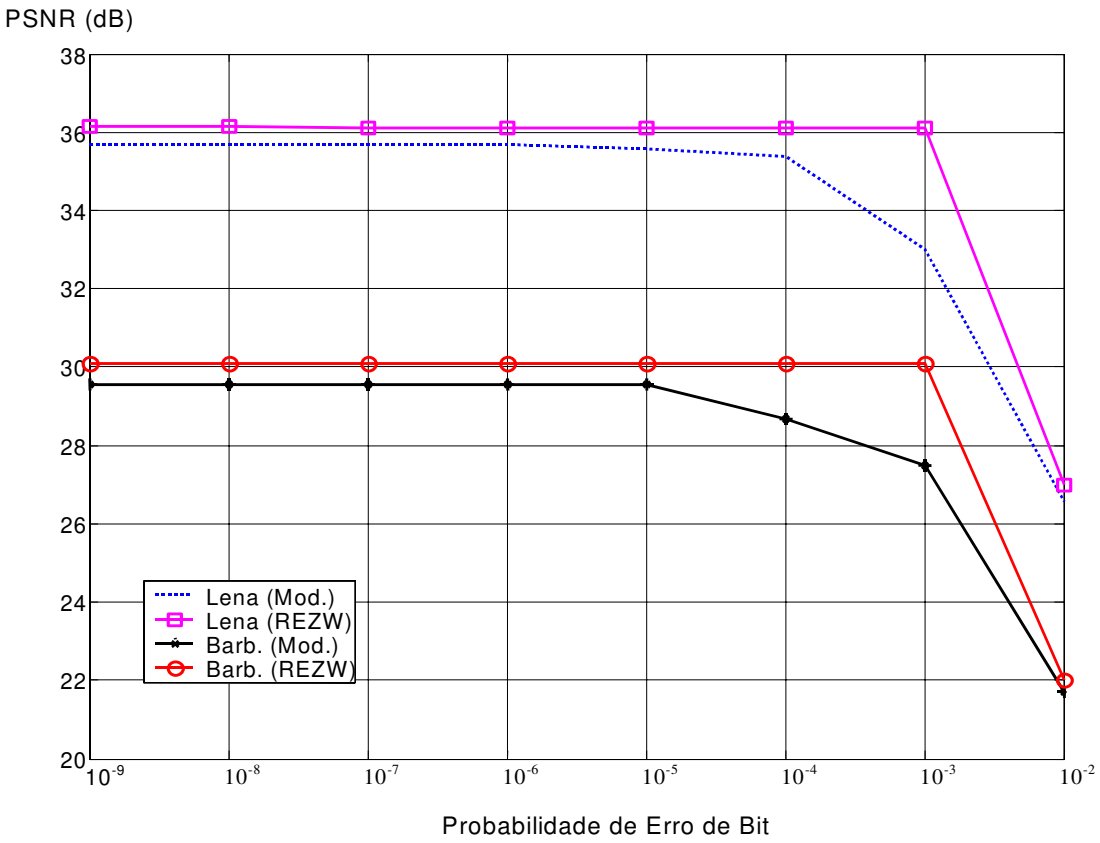


Fig. 4.9: PSNR × Probabilidade de Erro de Bit com S=256, codificada a uma taxa de 1bit/pixel usando divisão ZP.

(a) (b)

Fig.4.10: Imagem Lena codificada com o algoritmo modificado a uma taxa de 1 bit/pixel usando divisão ZP.

(a) S=256, Probabilidade de erro de bit de 10^{-2}, PSNR = 26,85 dB;
(b) S=1, Probabilidade de erro de bit de 10^{-2}, PSNR = 13,36 dB.

Para melhor visualização e comparação dos resultados as Fig. 4.11 e 4.12 apresentam as curvas do algoritmo modificado, codificado a uma taxa de 1 bit/pixel usando divisão ZP.

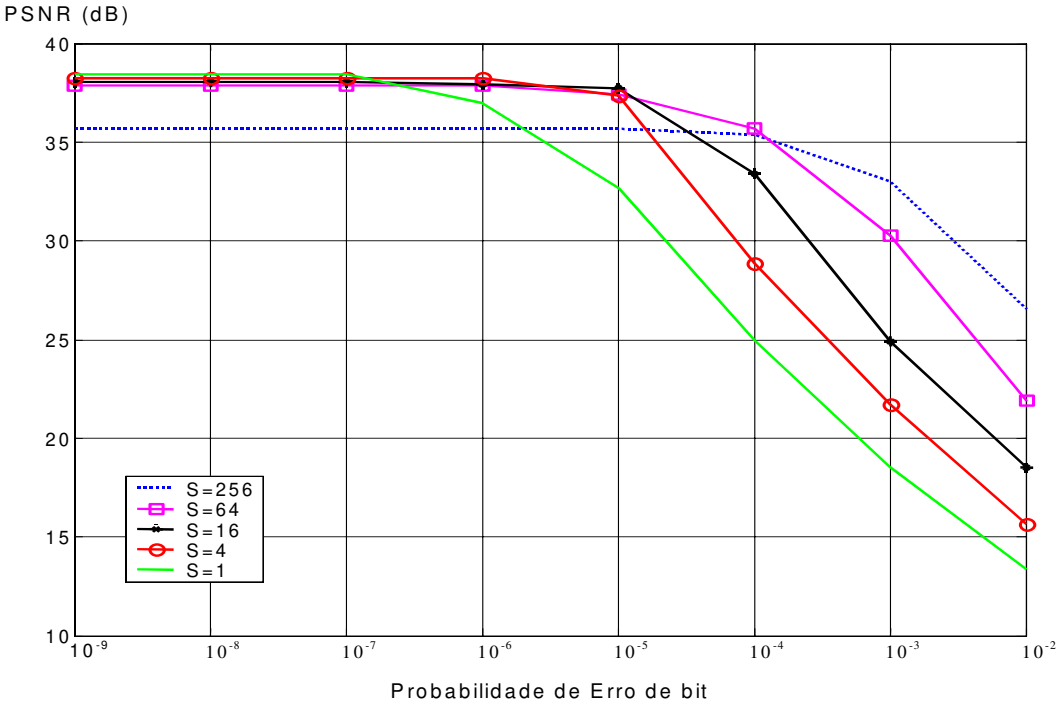


Fig.4.11: PSNR × Probabilidade de Erro de Bit para a imagem Lena.

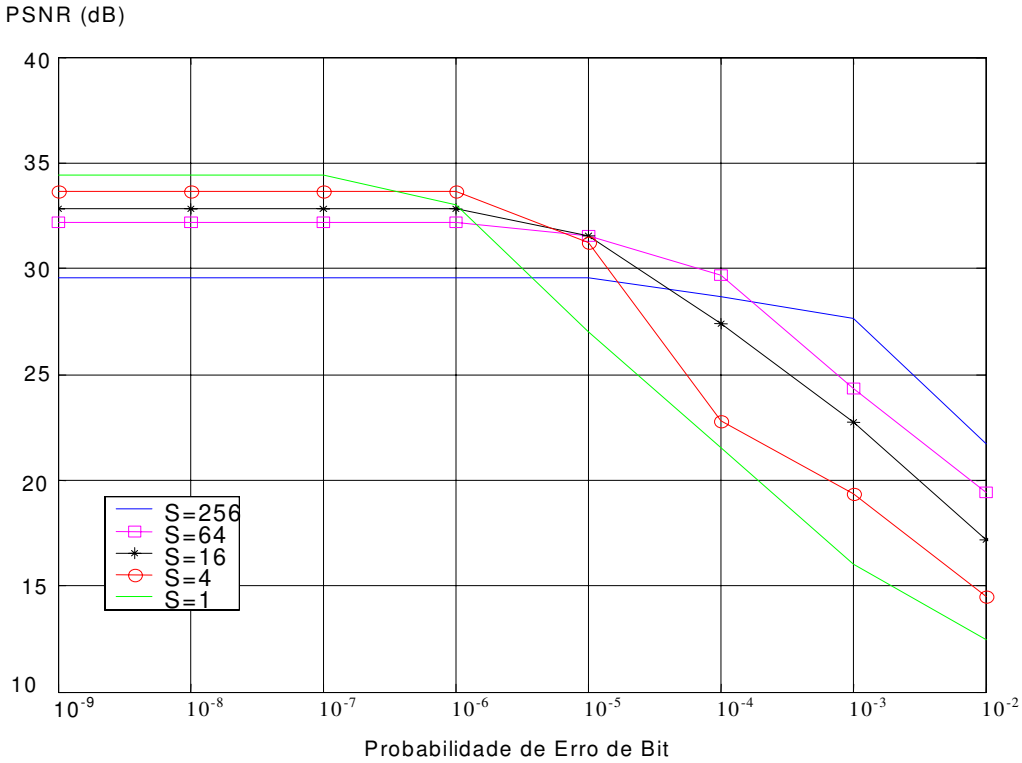


Fig. 4.12: PSNR × Probabilidade de Erro de Bit para a imagem Bárbara.

As Figs. 4.2 a 4.10 mostram os resultados das imagens Lena e Bárbara codificadas com os algoritmos modificado e REZW [33] a uma taxa de 1 bit/pixel. Em cada figura apresentam-se os resultados das duas imagens para diferentes números de seqüências S. Observa-se, para os diferentes valores de S, que os gráficos do algoritmo modificado e do algoritmo REZW não apresentam resultados iguais. Esta diferença de resultados deve-se ao fato do algoritmo REZW utilizar codificação aritmética enquanto o algoritmo modificado utiliza codificação de Huffman. Assim, o algoritmo REZW apresenta melhores resultados em relação ao algoritmo modificado (da ordem de 1,50 dB), mas ao custo de uma maior complexidade computacional.

Pode-se observar, através dos resultados objetivos e subjetivos apresentados, que uma maior robustez é alcançada pela codificação dos coeficientes em múltiplas seqüências antes da transmissão. À medida que o número de seqüências aumenta, a robustez do sistema também aumenta, particularmente para as probabilidades de erro mais altas, como pode ser observado nas Figs. 4.11 e 4.12. Por exemplo, para a imagem Lena, o PSNR obtido para a probabilidade de erro de bit de 10^{-2} é de 13,36 dB com S=1 e de 26,58 dB com S=256, para a taxa de 1 bit/pixel, demonstrando assim a maior robustez de um método em relação ao outro para uma mesma probabilidade de erro. Por outro lado, observa-se que o desempenho de PSNR × taxa de erro de bit é bastante prejudicado para as probabilidades de erro de bit menores que 10^{-6}, quando o número de seqüências aumenta, mostrando que a sobretaxa (*overhead*) exigida pelo algoritmo não é compensada para as baixas probabilidades de erro. Por exemplo, para a imagem Lena codificada a uma taxa de 1 bit/pixel, o PSNR obtido para a probabilidade de erro de 10^{-7} cai de 38,41 dB com S=1, para 35,69 dB com S = 256. A partir destas observações, fica clara a vantagem em se utilizar um número intermediário de seqüências, como, por exemplo, S=64. Neste caso, não perde muito em sobretaxa e, para as altas probabilidades de erro de bit, tem-se ainda uma alta robustez. Por exemplo, para a imagem Lena codificada a uma taxa de 1 bit/pixel com S = 64, o PSNR é de 37,87 dB para a probabilidade de erro de bit de 10^{-7}e de 21,89 dB para a probabilidade de erro de 10^{-2}, enquanto que para as mesmas probabilidades de erro, com S=1, os resultados obtidos são de 38,41 dB e de 13,36 dB, respectivamente.

Os resultados apresentados acima foram para a divisão de seqüências *ZP (zerotree partitioning*). A seguir apresentam-se os resultados comparativos para as divisões ZP e OZ (o*ffset zerotree*) codificadas com o algoritmo modificado.

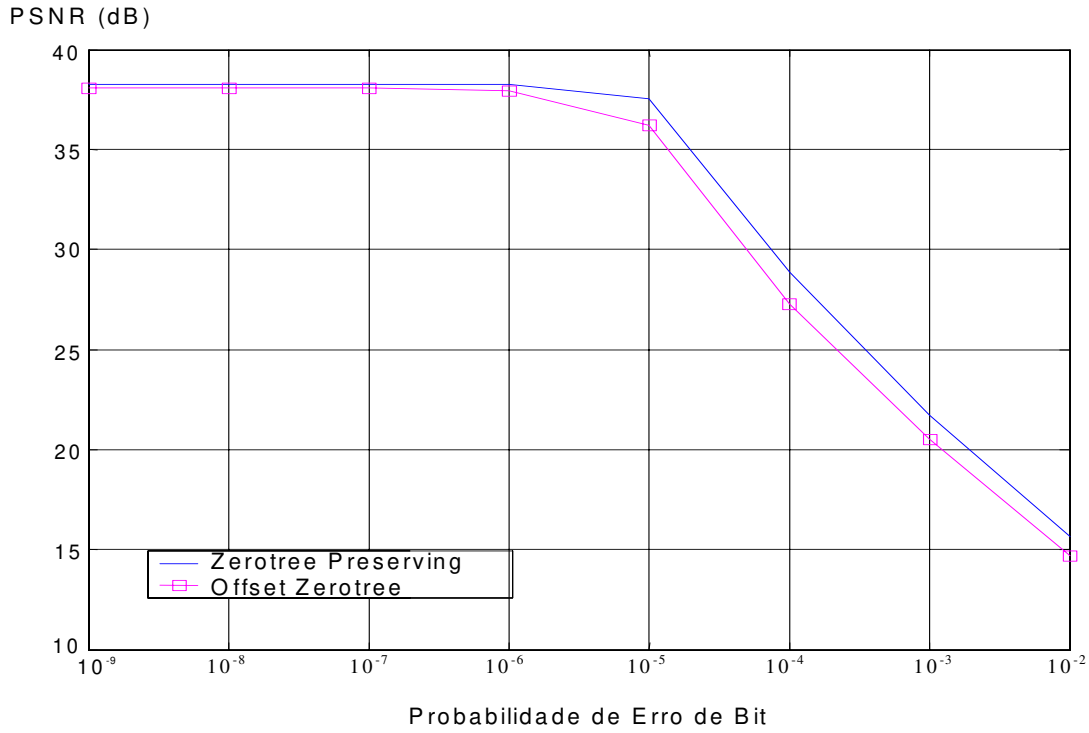


Fig. 4.13: PSNR × Probabilidade de Erro de Bit com S=4, para a Lena, codificada a uma taxa de 1bit/pixel.

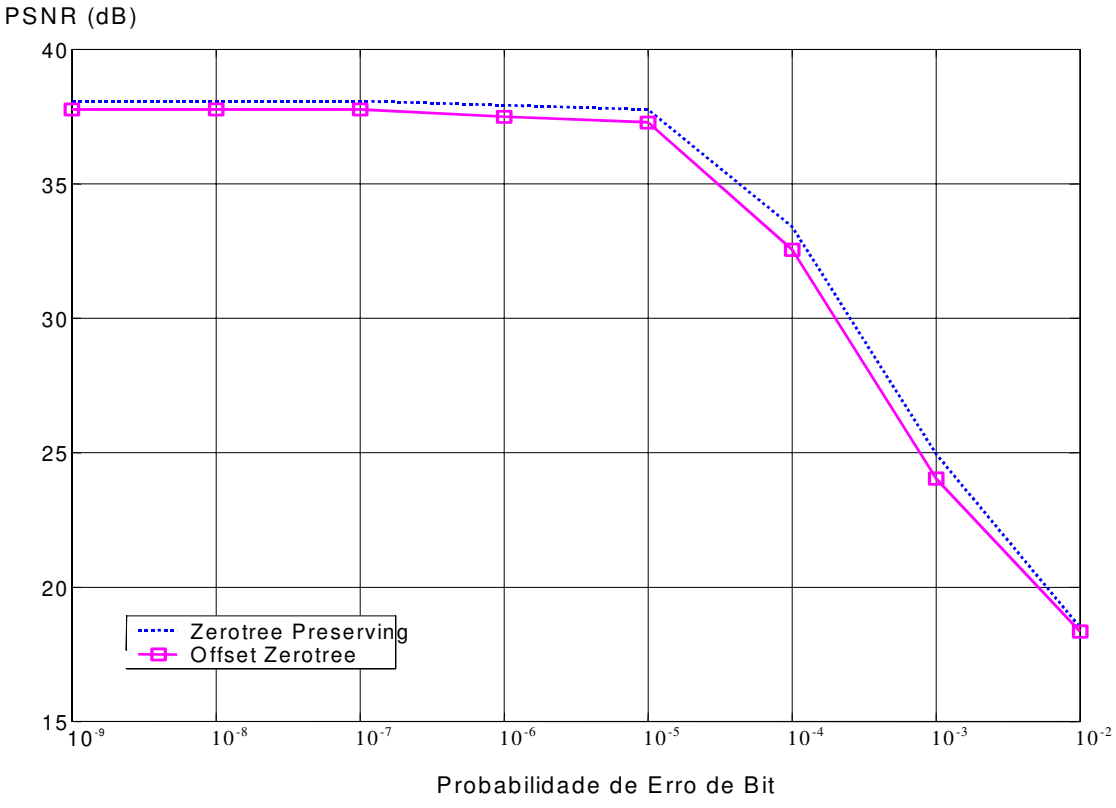


Fig. 4.14: PSNR × Probabilidade de Erro de Bit com S=16, para a Lena codificada a uma taxa de 1bit/pixel.

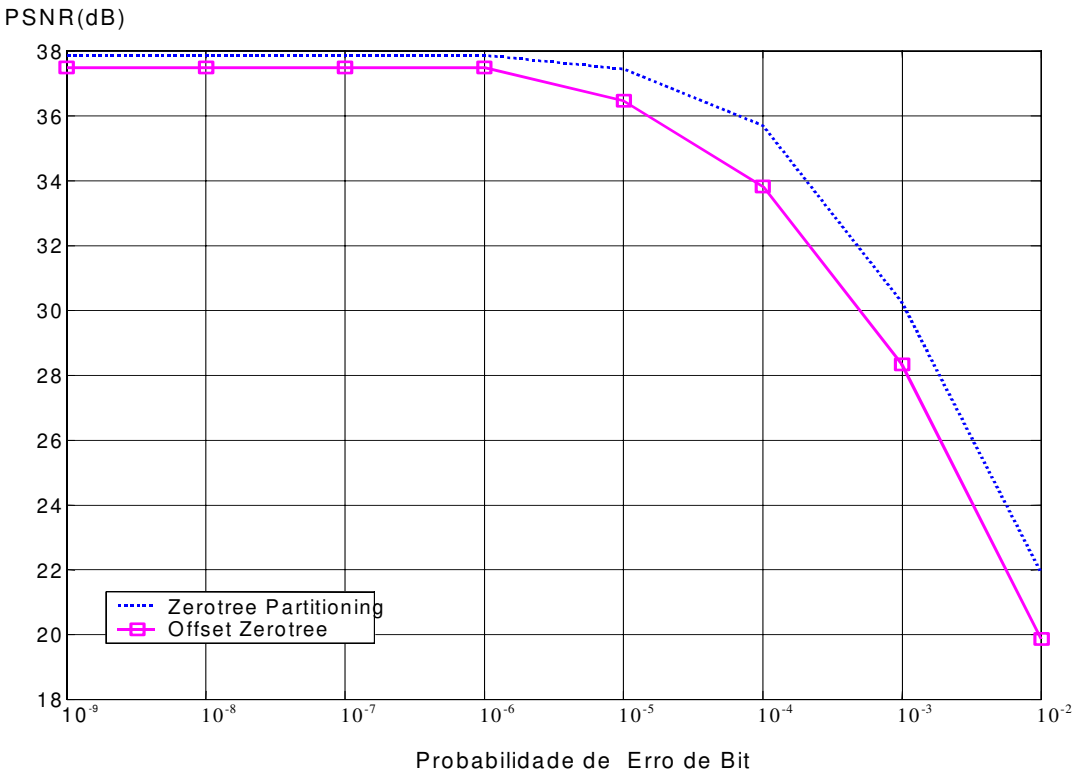


Fig. 4.15: PSNR × Probabilidade de Erro de Bit com S=64, para a Lena, codificada a uma taxa de 1bit/pixel.

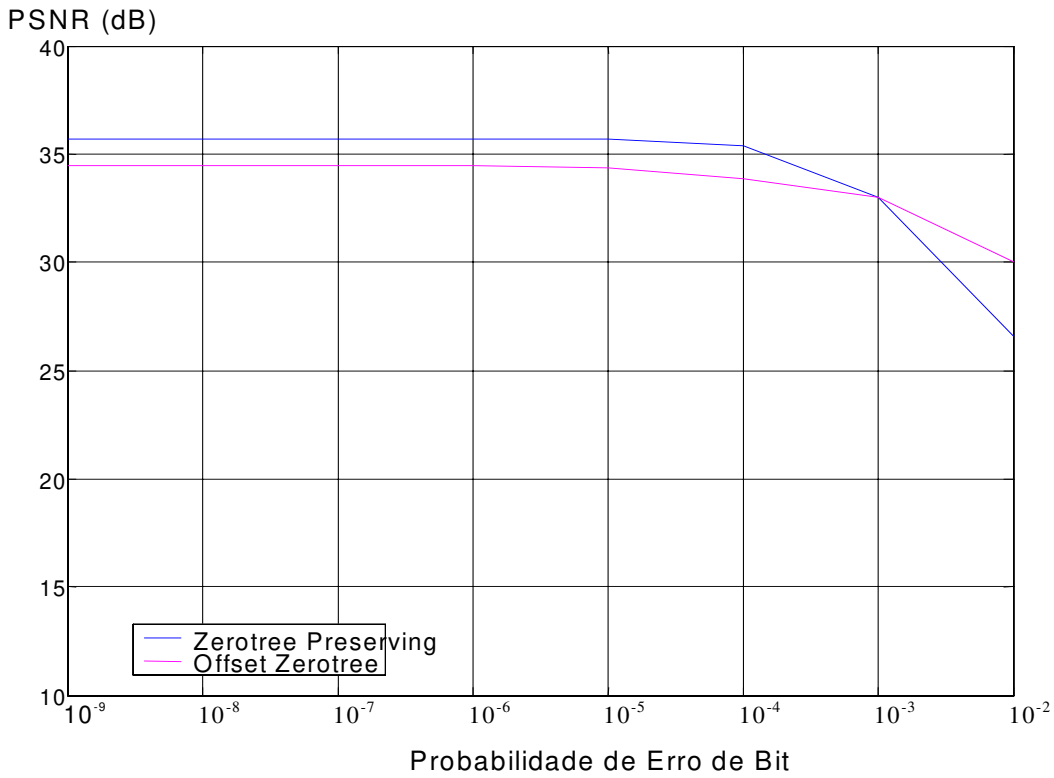


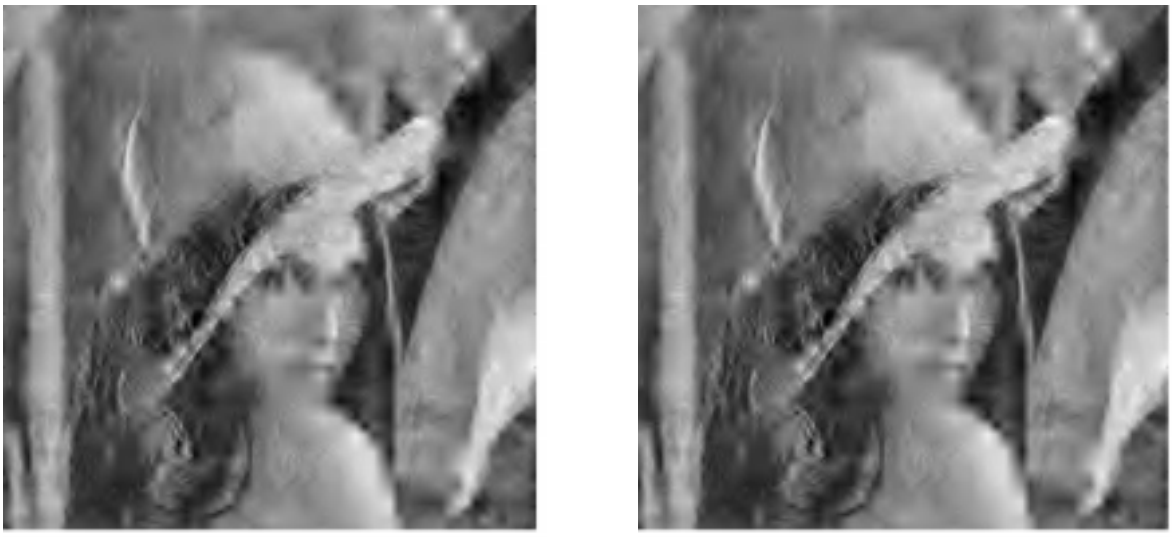
Fig. 4.16: PSNR × Probabilidade de Erro de Bit com S=256, para Lena codificada a uma taxa de 1bit/pixel.

A Fig. 4.17 apresenta os resultados subjetivos do algoritmo modificado para as divisões de seqüências ZP e OZ, ambas codificadas a uma taxa de 1 bit/pixel, com S = 256, e probabilidade de erro de bit igual a 10^{-3}.

(a) Divisão ZP, PSNR = 33,00 dB (b) Divisão OZ, PSNR = 31,83 dB

Fig. 4.17: Imagem Lena codificada a uma taxa de 1 bit/pixel, para S = 256 e probabilidade de erro de bit de 10^{-3}.

A figura a seguir esboça os resultados de reconstrução da imagem Lena em uma transmissão multicanal para as duas formas de divisão (ZP e OZ).



(a) (b)

Fig.4.18: Lena codificada a uma taxa de 1 bit/pixel, com S = 4, para uma transmissão multicanal com as probabilidades de erro de bit de 10^{-2}, 10^{-3}, 10^{-4} e 10^{-5} em cada seqüência.
(a) Divisão ZP (*Zerotree Partitioning*), PSNR= 20,03 dB;
(b) Divisão OZ (O*ffset Partitioning*), PSNR = 22,56 dB.

4.3 Algoritmo EZW e o Algoritmo Modificado

Nesta sessão comparam-se as eficiências do algoritmo modificado com apenas uma seqüência (S=1) e do codificador EZW proposto por Shapiro, para várias taxas de bit, i.e., para diferentes índices de compressão. Esta versão do algoritmo modificado com apenas uma seqüência (S=1) difere do algoritmo EZW apenas na codificação de entropia, que utiliza código de Huffman em vez de código aritmético. Portanto, o interesse desta comparação concentra-se no desempenho de PSNR × taxa de bit, pois, com S=1, ambos algoritmos são pouco robustos em relação a erros de transmissão. O código de Huffman, usado para o algoritmo modificado e para o algoritmo EZW não possui o símbolo de parada pois a imagem comprimida não será transmitida. Assim, somente os símbolos POS, NEG, IZ e ZTR são usados e só é medida a PSNR × taxa de bit.

A Fig. 4.19 ilustra a PSNR em função da taxa de bit para as imagens Lena e Bárbara, codificadas com os dois algoritmos de compressão (algoritmo modificado e algoritmo EZW).

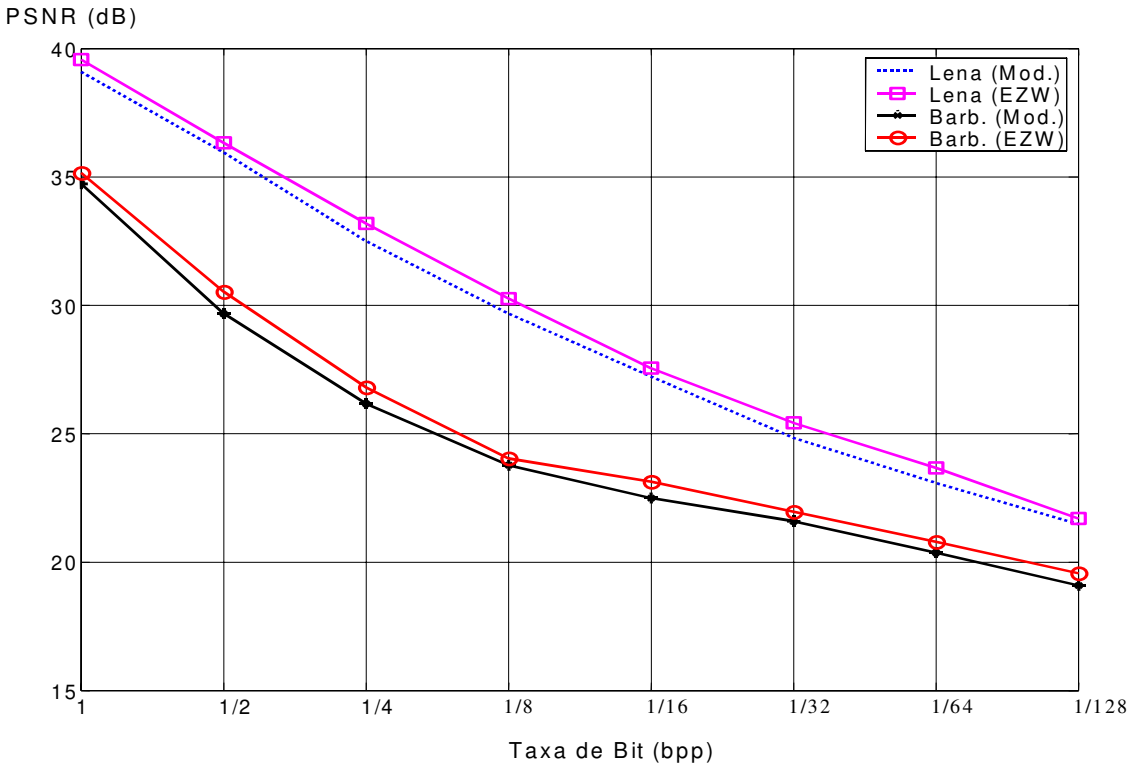


Fig. 4.19: Gráfico comparativo do algoritmo modificado e do algoritmo EZW[❖] para várias taxas de bit.

[❖] Os dados apresentados neste gráfico para o algoritmo EZW foram retirados de [Shapiro].

Para a imagem Lena, observa-se que para a taxa de 1 bit/pixel o codificador modificado apresentou PSNR = 39,05 dB, enquanto que, para o algoritmo EZW, o resultado foi de 39,55 dB [32]. Para a taxa de 1/128 = 0,0078 bits/pixel, o algoritmo modificado obteve PSNR igual a 21,47 dB e o algoritmo EZW obteve 21,69 dB. Resultados semelhantes são observados, para a imagem Bárbara. Há uma degradação de 0,5 dB da imagem Lena e Bárbara, ambas codificadas com o algoritmo modificado, em relação aos resultados obtidos com o algoritmo EZW. A diferença nos resultados dos dois algoritmos deve-se ao fato do codificador EZW utilizar codificação aritmética adaptativa, obtendo assim uma maior compressão, mas ao custo de uma maior complexidade computacional.

Para exemplificar a reconstrução obtida com o algoritmo modificado (com S=1), na Fig. 4.20 é esboçada a imagem Lena reconstruída com as taxas de 1bit/pixel e 1/16 = 0.0625 bits/pixel.

(a) (b)

Fig. 4.20: Desempenho do codificador modificado (com S=1) para a imagem Lena.
(a) 1 bit/pixel, compressão 8:1, PSNR=39,05 dB;
(b) 1/16 bits/pixel, compressão 128:1, PSNR=27,11 dB.

Capítulo 5

Conclusões

Neste trabalho propõe-se uma modificação do algoritmo REZW (*Robust Embedded Zerotree Wavelet*) proposto por Creusere [33]. Este algoritmo tem como base o algoritmo de compressão de imagens EZW (*Embedded Zerotree Wavelet*) inventado por Shapiro [32].

O codificador EZW foi especialmente projetado para trabalhar juntamente com a transformada *wavelet*. Ele realiza uma codificação progressiva, comprimindo uma imagem em uma seqüência de bits (*bitstream*). Esta seqüência permite a reconstrução da imagem com um aumento progressivo de resolução. Isto significa que quanto mais bits são adicionados à seqüência, mais detalhes a imagem reconstruída poderá conter. Codificação progressiva é também conhecida como codificação embutida (*embedded encoding*). O termo embutido é usado para indicar que os dados comprimidos são ordenados na ordem de importância visual. Assim, o codificador pode interromper a codificação quando uma desejada taxa de compressão for atingida ou alguma outra condição for satisfeita. Do mesmo modo, o decodificador pode interromper a decodificação a uma determinada taxa desejada. Esta característica é importante em termos de largura da banda utilizada, espaço de armazenamento, complexidade computacional, tempo e custo. Este método de compressão produz excelentes resultados sem requerer conhecimento à priori da imagem ou tabelas de codificação previamente armazenadas, além de não requerer nenhum treinamento dos codificadores e decodificadores.

No entanto, a imagem codificada com o algoritmo EZW apresenta pouca robustez a erros de bits quando transmitida em um canal ruidoso. Assim, Creusere [33] propôs um algoritmo, baseado no codificador EZW, que alcança uma maior robustez a erros de bits na transmissão. Este algoritmo denomina-se REZW (*Robust Embedded Zerotree Wavelet*). Sua robustez é alcançada pela divisão dos coeficientes transformados em grupos que são independentemente processados usando um codificador embutido. Deste modo, quando ocorre um erro de bit em um dos grupos, os bits dos outros grupos não são afetados, permitindo que mais informação correta chegue ao decodificador.

Este algoritmo foi especialmente implementado para ser usado em canais de comunicação sem fio, onde há uma maior necessidade de proteção contra erros durante a transmissão. Recentemente, a proliferação de serviços sem fio e da Internet, juntamente com a demanda por produtos de multimídia, tem estimulado o interesse na transmissão de imagens e de vídeo em canais ruidosos de comunicação cuja capacidade pode variar com o tempo.

A codificação realizada é uma codificação conjunta de fonte e canal que faz com que a curva de desempenho (Probabilidade de Erro de Bit x PSNR) tenha uma degradação suave à medida que a probabilidade de erro aumenta (cf. Figs 4.11 e 4.12). Se uma codificação de canal adicional for usada (e.g., codificação Reed Solomon), a curva de desempenho pode se manter estável até uma determinada probabilidade de erro, e em seguida, ter um decaimento abrupto quando a capacidade de correção do código de canal for excedida.

O algoritmo proposto neste trabalho é uma versão modificada do algoritmo EZW Robusto (REZW) e é denominado algoritmo modificado. Este algoritmo utiliza um codificador de Huffman em vez do codificador aritmético adaptativo utilizado no algoritmo REZW [33]. O algoritmo modificado apresenta resultados um pouco inferiores aos resultados apresentados pelo algoritmo REZW, da ordem de 1 a 3 dB, como é visto no Capítulo 4. Por exemplo, para a imagem Lena codificada com S=16, o PSNR obtido para a probabilidade de erro de 10^{-7} é de 38,65 dB para o algoritmo REZW e de 38,05 dB para o algoritmo modificado. Para a probabilidade de erro de 10^{-2} a PSNR cai para 19,00 dB para o algoritmo REZW e para 18,50 dB para o algoritmo modificado. As diferenças de resultados do algoritmo REZW com o algoritmo modificado se dão pelo uso do codificador de Huffman pelo algoritmo modificado em vez do codificador aritmético, mas com a vantagem de maior simplicidade computacional. Além do mais, o codificador aritmético tem o inconveniente de ser protegido por patentes, enquanto o codificador de Huffman é de domínio público.

Pode-se observar através dos resultados apresentados no Capítulo 4, que o algoritmo de compressão de imagens modificado apresenta robustez a erros de bits. Esta robustez pode ser incorporada com um pequeno custo de complexidade, i. e., pela divisão do vetor de bits em seqüências independentes antes da transmissão. À medida que o número de seqüências independentes aumenta, a robustez a erros de bits do codificador também aumenta, mas o desempenho taxa × distorção do codificador-decodificador decresce quando não há erros de bits. Por exemplo, para a imagem Lena codificada a uma taxa de 1 bit/pixel, o PSNR obtido para a probabilidade de erro de 10^{-7} cai de 38,41 dB com apenas uma seqüência (S=1), para

35,69 dB com S = 256. Se a taxa de erro de bit é alta, entretanto, o desempenho deste algoritmo é geralmente melhor que o do algoritmo EZW para valores de S maiores que um. Por exemplo, para a imagem Lena codificada a uma taxa de 1 bit/pixel, o PSNR obtido para a probabilidade de erro de bit de 10^{-2} é de 13,36 dB, com S=1 e de 26,58 dB, com S=256. Resultados semelhantes são encontrados para a imagem Bárbara.

Através dos resultados apresentados no Capítulo 4 para os diferentes números de seqüências, fica clara a vantagem em se utilizar um número de seqüências intermediário, mantendo assim a robustez do sistema quando transmitido em canais ruidosos, e não perdendo muito em sobretaxa, que seria desnecessária, para a transmissão em um canal com baixa probabilidade de erro. Por exemplo, para a imagem Lena codificada a uma taxa de 1 bit/pixel, com S=64 seqüências, o PSNR é de 37,87 dB para a probabilidade de erro de bit de 10^{-7} e de 30,24 dB para probabilidade de erro de 10^{-3}.

No Capítulo 4 são apresentados também alguns resultados da divisão de seqüências OZ (*Offset Zerotree*) comparadas com os resultados da divisão de seqüências ZP (*Zerotree Partitioning*). A partir dos resultados apresentados, observa-se uma ligeira superioridade da divisão ZP em relação a divisão OZ (da ordem de 0,50 dB) para as seqüências menores que S=256. Com S=256, para baixas probabilidades de erro (até 10^{-5}), a divisão ZP apresenta resultados superiores. Entretanto, para as altas probabilidades de erro, a divisão ZP apresenta resultados inferiores em relação à divisão OZ com S=256. A superioridade dos resultados da divisão ZP em relação aos resultados da divisão OZ acontece porque a divisão de seqüências OZ não preserva a relação de dependência da estrutura *zerotree* convencional. Em vez disto, esta estrutura de descendência se distribui entre as escalas *wavelets* de forma que nenhum coeficiente adjacente é colocado no mesmo grupo. Quando um vetor de bits é terminado prematuramente por erro de transmissão, os coeficientes de resolução reduzida correspondentes estarão rodeados pelos coeficientes de resolução completa, que foram transmitidos em seqüências independentes. O que explica a divisão OZ ter tido resultados superiores para as altas probabilidades de erro com S=256.

As imagens codificadas pelas divisões ZP e OZ também foram transmitidas em um ambiente de múltiplos canais com diferentes probabilidades de erro de bit, com S=4 seqüências. Observa-se que a divisão OZ apresentou um resultado superior ao da divisão ZP (2,53 dB) para as mesmas condições de erro de canal.

Através dos resultados apresentados, conclui-se que o algoritmo modificado, apesar de ser um método bastante simples em relação ao codificador EZW, apresenta alguma robustez mesmo quando transmitido em um canal com alta probabilidade de erro, preservando assim a qualidade da imagem reconstruída.

Uma possível extensão do presente trabalho é o desenvolvimento de um algoritmo robusto que tenha como base o algoritmo SPIHT [35], ao invés do algoritmo EZW, como foi utilizado neste trabalho. Uma outra sugestão para trabalhos futuros seria o uso de um algoritmo adaptativo, onde o número de seqüências varia de acordo com a probabilidade de erro do canal. Neste caso, para um canal com alta probabilidade de erro, utilizar-se-ia um maior número de seqüências, e para um canal pouco ruidoso, um número de seqüências pequeno, diminuindo assim a sobretaxa imposta pelo algoritmo.

Apêndice A

O Algoritmo SPIHT (*Set Partitioning in Hierarchical Trees*)

O algoritmo SPIHT (*Set Partitioning in Hierarchical Trees*) proposto por Said e Pearlman [35], é uma extensão do algoritmo EZW [32]. Neste algoritmo, a forma como os subconjuntos de coeficientes são divididos e como a informação é transmitida são fundamentalmente diferentes do algoritmo EZW. Os resultados obtidos por este método são melhores que os resultados obtidos pelo algoritmo EZW ou de métodos de codificação muito mais sofisticados e complexos, como quantização vetorial codificada por treliças. Há ainda a vantagem de ser extremamente rápido.

No algoritmo EZW original [32], a codificação aritmética foi essencial para comprimir a informação. No SPIHT, a divisão dos subconjuntos é tão eficiente e a informação significativa tão compacta que mesmo uma transmissão não codificada em entropia alcança o mesmo ou melhor desempenho que o algoritmo EZW. Com a utilização do codificador aritmético, o erro quadrático médio pode ainda diminuir, ou a medida PSNR aumentar de 0,3 a 0,6 dB para uma mesma taxa ou tamanho de arquivo comprimido. O código transmitido ou arquivo da imagem comprimida é completamente embutido (*embedded*), tal que um arquivo de uma imagem comprimida em uma dada taxa pode ser truncado em vários pontos e decodificado para formar uma série de imagens reconstruídas em taxas de bits mais baixas. O algoritmo EZW não pode obter seu melhor desempenho em todas as taxas a partir de um único arquivo. Para cada taxa, o algoritmo EZW precisa otimizar um certo parâmetro. Por outro lado, o algoritmo SPIHT resolve este problema pela variação da priorização da transmissão, e produz com um único arquivo embutido, seu melhor desempenho para todas as taxas. A codificação pode parar em um tamanho arbitrário do arquivo comprimido ou continuar até que o arquivo comprimido seja uma representação aproximada (quase sem perda) da imagem. “Quase sem perda” porque a compressão não é reversível, pois os filtros escolhidos para a transformada *wavelet* têm *taps* não inteiros e estes produzem coeficientes transformados não inteiros, os quais são aproximados para uma precisão finita [40].

Árvores de Orientação Espacial

Uma estrutura em árvore, chamada árvore de orientação espacial, define a relação espacial na pirâmide hierárquica, como no algoritmo EZW. A Fig. A.1 mostra como a orientação espacial da árvore é definida. Cada nó da árvore corresponde a um pixel, e é identificada pela coordenada deste pixel. A árvore é definida de tal forma que cada nó tenha nenhum ou 4 descendentes, os quais sempre formam um grupo de pixels adjacentes 2x2. Seus descendentes diretos (seus filhos) correspondem aos pixels de mesma orientação espacial no próximo nível mais fino da pirâmide. Na figura, as setas são orientadas na direção dos pais para os 4 filhos. Os pixels no nível mais alto da pirâmide são as raízes da árvore e são também agrupados em pixels adjacentes 2x2, como no algoritmo EZW. Entretanto, a forma como seus descendentes se ramificam é diferente do algoritmo EZW. Em cada grupo da raiz da árvore, um dos pixes (indicado pelo círculo na figura), não tem descendente.

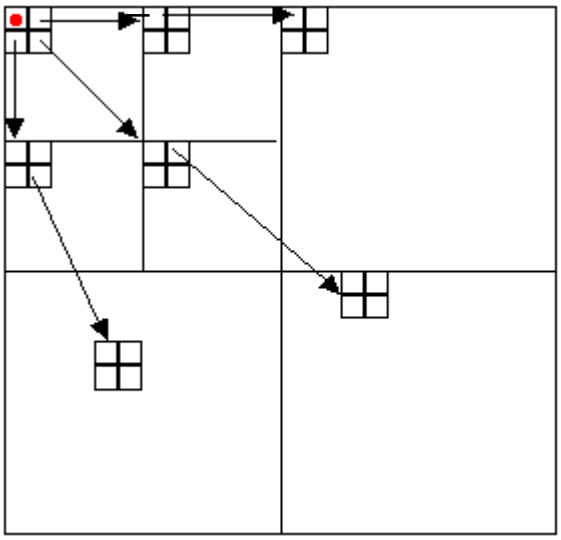


Fig. A1: Árvore de orientação espacial
da dependência entre pai-filho.

Os seguintes conjuntos de coordenadas são usados para apresentar o novo método de codificação [35]:

- $\boldsymbol{O}(i,j)$: conjunto com as coordenadas de todos os filhos do nó (i,j);
- $\boldsymbol{D}(i,j)$: coordenadas de todos descendentes do nó (i,j);
- $\boldsymbol{H}$: coordenadas de todas as raízes da árvore (nós no nível mais alto da pirâmide);

• $\boldsymbol{L}(i,j) = \boldsymbol{D}(i,j) - \boldsymbol{O}(i,j)$: coordenadas de todos os descendentes do nó (i,j) exceto a de seus filhos.

Exceto para os níveis mais altos e mais baixos da pirâmide , tem-se o seguinte conjunto de coordenadas de todos os filhos do nó (i,j)

$$\boldsymbol{O}(i,j) = \{(2i,2j),(2i,2j+1),(2i+1,2j),(2i+1,2j+1)\}. \quad \text{(A.1)}$$

As regras da divisão do conjunto são as seguintes:

1 – Inicialmente tem-se os conjuntos $\{(i,j)\}$ e $\boldsymbol{D}(i,j)$, para todo $(i,j) \in \boldsymbol{H}$;
2 – Se $\boldsymbol{D}(i,j)$ é significante então ele é dividido em $\boldsymbol{L}(i,j)$ e em mais quatro conjuntos de um elemento com $(k,l) \in \boldsymbol{O}(i,j)$.
3 – Se $\boldsymbol{L}(i,j)$ é significante, ele é dividido em quatro conjuntos $\boldsymbol{D}(k,l)$, com $(k,l) \in \boldsymbol{O}(i,j)$.

O Algoritmo de Codificação SPIHT

Uma das principais características proposta pelo algoritmo SPIHT é que os dados não são explicitamente transmitidos. Uma outra característica é o fato de não se precisar selecionar todos os coeficientes. É necessário um algoritmo que simplesmente selecione os coeficientes tal que $2^n \leq |c_{i,j}| < 2^{n+1}$, com n decrementado em cada passo, como na EZW. Dado n, se $|c_{i,j}| \geq 2^n$, então o coeficiente é significante, caso contrário é insignificante.

O algoritmo SPIHT usa dois tipos de símbolos: O símbolo "1" para indicar que o coeficiente é significante e "0" para coeficientes insignificantes. Neste caso, a lista dominante usada no algoritmo EZW original, para armazenar os coeficientes insignificantes, é substituída por duas listas, Lista de Pixels Insignificantes (LIP) e Lista de Conjuntos Insignificantes (LIS). A Lista de Pixels Insignificantes (LIP), é usada para visitar e examinar os coeficientes insignificantes individualmente. Enquanto a Lista de Conjuntos Insignificantes (LIS), é usada para examinar os conjuntos de descendentes insignificantes dos coeficientes da árvore. Se Γ representa um conjunto de coeficientes, então os símbolos usados para codificar o mapa de significância são mapeados da seguinte forma [35]:

$$S(\Gamma, T_i) = \begin{cases} 1, & se \quad \max_{m \in \Gamma}(|\, c_m \,|) \geq T_i, \\ 0, & caso \quad contrário. \end{cases} \tag{A.2}$$

É importante notar que um coeficiente em particular pode fazer parte de ambas as listas (LIP e LIS), se ambos, o nó (*i,j*) e seus descendentes são insignificantes, o que é equivalente a ter uma *zerotree* na EZW original. Conseqüentemente, a passagem dominante do algoritmo EZW é substituída por duas sub-passagens: uma sub-passagem é realizada para visitar os coeficientes na lista LIP, e a outra para visitar os conjuntos na lista LIS. Os coeficientes classificados como significantes durante as duas sub-passagens são inseridos em uma terceira lista (Lista de Pixels Significantes – LSP). Os sinais dos coeficientes classificados como significantes são transmitidos.

Outra característica distinta do algoritmo SPIHT é a forma hierárquica de construção de suas listas. Por exemplo, em vez de listar todos os coeficientes como membros da lista dominante e iniciá-los como nós de uma *zerotree*, como feito no algoritmo EZW, somente as raízes principais da árvore (coeficientes no nível mais alto) são inseridas nas listas LIS e LIP. O conjunto de raízes inclui todos os coeficientes da subbanda de nível mais alto (subbanda DC), exceto o coeficiente do topo superior esquerdo (o coeficiente DC).

A transmissão da imagem é realizada de forma progressiva como na EZW, ou seja, os coeficientes com maior magnitude devem ser transmitidos primeiro, pois eles contêm maior informação da imagem. Isto corresponde ao método de transmissão proposto por Devore [41].

Apêndice B

Imagens Utilizadas nas Simulações

Fig. B.1: Lena Original 512x512

Fig. B.2: Bárbara Original 512x512

Referências Bibliográficas

[1] M. Antonini, M. Barlaud, P. Mathieu e I. Daubechies, "Image Coding Using Wavelet Transform". IEEE Trans. On Image Processing, vol.1, no 2, Abril 1992.

[2] J. S. Lim, *Two- Dimensional Signal and Image Processing*. Englewood Cliffs, N.J., Prentice Hall, 1989.

[3] A. K. Jain, *Fundamental of Digital Image Processing*. Englewood Cliffs, N.J., Prentice Hall, 1988.

[4] G. Strang e T. Najuyen, *Wavelets and Filter Banks*. Wellesley – Cambridge Press, 1996.

[5] N. S. Jayant, *Waveform Quantization and Coding*. New York: IEEE Press, 1976.

[6] N. Abramson, *Information Theory and Coding*, Mc Graw Hill Book Company, 1963.

[7] G. G. Langdon, "An Introduction to Arithmetic Coding", IBM J. Res. Dev., vol.28, no 2, pag. 135-149, 1984.

[8] J. Rissanen e G. G. Langdon, "Arithmetic Coding", IBM J. Res. Dev., vol. 23, no 2, pag. 149-162, 1979.

[9] O Rioul e M. Vetterli, "Wavelets and Signal Processing". IEEE Signal Processing Magazine, vol.8, no 4, pag. 14-38, Outubro 1991.

[10] P.A. Morettin, "Ondaletas e Seus Usos na Estatística". In: Escola de Séries Temporais e Ecometria (Canela - Brasil), Agosto 1997.

[11] J. Daubechies, *Ten Lectures on Wavelets*, Philadelphia, Pensylvania. Siam, 1992.

[12] C. F. Chui, *An Introduction to Wavelets*. San Diego . Academic Press Inc., 1992.

[13] M. Vetterli e J. Kovacevic, *Wavelets and Subband Coding*,. Englewood Cliffs, N.J., Prentice Hall, 1995.

[14] C. S. Burrus, R.A. Gonipath e H. Guo, *Introduction to Wavelet and Wavelet Transforms*. A Primer. Englewoods Cliffs, N.J., Prentice Hall, 1997.

[15] A. Cohen, I. Daubechies e J. C. Feauveau, "Biorthogonal Bases of Compactly Supported Wavelets". Communications on Pure and Applied Mathematics, vol. 45, pag. 485-560, 1992.

[16] B. Porat, *Digital Processing of Randon Signal – Theory and Methods,* Cap. 11, Englewood Cliffs, INC., Prentice Hall, 1994.

[17] S. G. Mallat, “A theory for Multiresolution Signal Decomposition: The Wavelet Representation”, IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 11, no 7, Julho 1989.

[18] P. P. Vaiadynathan, “Quadrature Mirror Filter Banks, M-Band Extensions and Perfect-Reconstruction Techiniches”, IEEE ASSP Magazine, pag. 4-20, Julho 1987.

[19] P. J. Burt e E. H. Adelson, “The Laplacian Pyramid as a Compact Image Code”. IEEE Trans. Communications, vol.31, no 4, pag. 532-540, Março 1983.

[20] J. Crowley, “A Representation for Visual Information”, Robotic Inst. Carnegie-Mellon Univ., Tech. Rep. CMU-RI-TR-82-7, 1987.

[21] T. J. M. Smith e S. L. Eddins, “Subband Coding of Imagens wiht Octave Band Tree Struture”, Procedings IEEE International Conference on Acoustic, Speech and Signal Processing, pag. 1382-1385, Abril 1987.

[22] Y. T. Chan, *Wavelet Basics*, Kluwer Academic Publishers, 1995, Cap.II, pag. 23-51.

[23] K. R. Castleman, *Digital Image Processing*, 2ed, Mc. Graw Hill, 1995.

[24] O. Egger e W. Li, “Subband Coding of Images Using Asymmetrical Filter Banks”, IEEE Trans. on Image Processing, vol.4, no 4, Abril 1995.

[25] A. Gersho e R. M. Gray, “Vector Quantization and Signal Compression”, Kluwer Academic Publishers, 1991.

[26] S. P. Lloyd, “Least Squares Quantization in PCM”, IEEE Trans. on Information Theory, vol IT-28, pag. 129-137, Março 1982.

[27] J. Max, “Quantization for Minimun Distortion”, IEEE Trans. on Information Theory, pag. 7-12, Março 1960.

[28] T. M. Cover e J. A. Thomas, *Elements of Information Theory*, New York, Wiley 1991.

[29] J. H. Witten, R. M. Neal e J. G. Cleary, “Arithmetic Coding for Data Compression”, Comm. of ACM, vol.30, no 6, pag. 520-540, 1987.

[30] J. W. Woods e S. D. O’Neil, “Subband Coding of Images”, IEEE Transactions on Acoustics, Speech and Signal Processing, vol. ASSP-34, no 5, Outubro 1986.

[31] H. Gharavi e A. Tabatabai, “Subband Coding of Monochrome and Color Images”, IEEE Transactions on Circuits and Systems, vol. 35, no 2, pag. 207-214, Fevereiro 1988 .

[32] J. M. Shapiro, “Embedded Image Coding Using Zerotrees of Wavelets Coefficientes”, IEEE Transactions on Signal Processing, vol. 41, no 12, Dezembro 1993.

[33] C. D. Creusere, "A New Method of Robust Image Compression Based on the Embedded Zerotree Wavelet Algorithm", IEEE Transactions on Image Processing, vol. 6, no 10, Outubro 1997.

[34] D. Taubman, "High Performance scalable image compression with EBCOT", IEEE Trans. Image Processing, vol. 9, Julho 2000.

[35] A. Said e W. A. Pearlmman, "A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees", IEEE Transactions on Circuits and Systems for Video Technology, vol. 6, no 3, Junho 1996.

[36] M. Rabbani e P. W. Jones, *Digital Image Compression Techiniques*. SPIE Optical Engeneering Press, 1991.

[37] B. E. Usevitch, "A Tutorial on Modern Lossy Wavelet Image Compression: Foundations of JPEG 2000", IEEE Signal Processing Magazine, pag. 22-35, Setembro 2001

[38] A. B. Carlson, *Communication Systems*, Mc. Graw Hill, New York, NY, USA, 1986, 3ed, pag. 419-423.

[39] T. Berger, *Rate Distortion Theory: A Mathematical Bases for Data Compression*, pag. 149. Prentice Hall, Englewood Cliffs, NJ, 1971

[40] J. D. Villasenor, B. Belzer e J. Liao, "Wavelet Filter Evaluation for Image Compression", IEEE Transactions on Image Processing, vol. 4, no 8, Agosto 1995.

[41] R. A. Devore, B. Jawerth e B. J. Lucier, "Image Compression Through Wavelet Transform Coding", IEEE Transactions Information Theory, vol. 38, pag. 719-746, Março 1992.