IONILDO JOSÉ SANCHES

COMPRESSÃO SEM PERDAS DE PROJEÇÕES DE TOMOGRAFIA COMPUTADORIZADA USANDO A TRANSFORMADA WAVELET

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Curso de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.

Orientador: Prof. Eduardo Parente Ribeiro

CURITIBA

2001

Ministério da Educação
Universidade Federal do Paraná
Mestrado em Informática

PARECER

Nós, abaixo assinados, membros da Banca Examinadora da defesa de Dissertação de Mestrado em Informática do aluno ***Ionildo José Sanches***, avaliamos o trabalho intitulado ***"Compressão sem perdas de projeções de tomografia computadorizada usando a transformada Wavelet"***, cuja defesa foi realizada no dia 20 de fevereiro de 2001. Após a avaliação, decidimos pela aprovação do candidato.

Curitiba, 20 de fevereiro de 2001.

Prof. Dr. Eduardo Parente Ribeiro
Presidente

Prof. Dr. Jacques Facon
Membro Externo – PUC-PR

Profª. Dra. Olga Regina Pereira Bellon
DINF/UFPR

AGRADECIMENTOS

Agradeço primeiramente a Deus por iluminar os caminhos em minha vida na conquista de novos desafios.

Agradeço de forma especial a meus pais e a minha irmã pelo amor, dedicação e apoio que sempre me deram durante a realização de todo este trabalho. A eles sou eternamente grato.

Ao professor Eduardo Parente Ribeiro, meu orientador, pela enorme contribuição dada para o desenvolvimento deste trabalho.

Ao Hamilton, do LAC, que mostrou todo o processo de aquisição de imagens de tomografia computadorizada e gentilmente cedeu algumas imagens que permitiram a realização dos experimentos.

Agradeço ainda a todos meus colegas do curso de pós-graduação em informática, os professores e a todos aqueles que de alguma forma deram a sua contribuição.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

RESUMO

Este trabalho apresenta o uso da transformada *wavelet* para compressão sem perdas de projeções de tomografia computadorizada. A compressão de imagens digitais é essencial para aplicações tais como armazenamento e transmissão. Métodos de compressão de imagens com perdas podem alcançar altas taxas de compressão, porém impedem a reconstrução exata dos dados originais. Os dados referentes às projeções de tomografia, chamadas senogramas, não podem sofrer perdas no processo de compressão e descompressão para não afetar a reconstrução da imagem. Portanto, será utilizada a transformada *wavelet* de inteiros reversível que permite realizar a compressão sem perdas.

ABSTRACT

This work presents how wavelet transform can be used for lossless compression of computerized tomography projections. Digital image compression is essential for applications such as storage and transmission. Lossy image compression methods can achieve high compression ratios, but they don't allow exact reconstruction of the input data. Tomography projections data, called sinograms, should not loose information during compression and decompression process to not affect the image reconstruction. Thus, the reversible integer wavelet transform which allows lossless image compression will be used.

1. Introdução

A utilização da tomografia computadorizada (*Comput(eriz)ed Tomography* - CT) tornou-se um elemento indispensável em hospitais e clínicas de todo o mundo. Nos últimos anos a CT tem se estendido também para aplicações industriais. A partir das projeções, obtidas com as medidas realizadas externamente a partir de diferentes ângulos de um corpo, deve-se reconstruir esses dados originais utilizando alguma técnica de reconstrução. Existem vários algoritmos e técnicas que permitem a reconstrução da imagem a partir das projeções obtidas [26, 28, 41].

As imagens digitais geram um grande volume de dados, o que torna evidente a necessidade de aplicar alguma técnica de compressão de dados, que permita reduzir a quantidade de informação necessária, garantindo a exatidão dos dados e facilitando o armazenamento, a transmissão e o processamento desses dados.

Por exemplo, uma única projeção com 768 × 90 pontos (ou *pixels*) com 65.536 níveis de cinza (16 bits por *pixel*) necessita de 138.240 bytes para ser armazenada. Esta mesma imagem após a sua reconstrução necessitará de 1.179.648 bytes se mantido os 16 bits por *pixel*. A transmissão de imagens de CT também é outro fator que deve ser levado em consideração, já que cada vez mais a transmissão de imagens se torna necessária. Em vista disso, as técnicas de compressão têm a finalidade de diminuir esta quantidade de dados, mantendo a integridade dos dados originais que poderão ser reconstruídos a qualquer instante que se faça necessário. Em vista dessa grande quantidade de informação, a compressão de dados é um passo inevitável na construção de sistemas de arquivamento e comunicação de imagens (*Picture Archiving and Communications Systems* – PACS) e aplicações de

telemedicina que requerem o armazenamento e a transmissão de uma grande quantidade de dados.

Através do processo de compressão de dados podemos reduzir a quantidade de dados necessários para representar uma determinada quantidade de informação. Vários métodos de compressão de dados sem perdas (reversíveis) e com perdas (irreversíveis) têm sido propostos na literatura nos últimos anos. Cada um desses métodos procuram explorar determinadas características presentes na imagem. A escolha do método apropriado depende do conhecimento da teoria do método, bem como, um prévio conhecimento das características presentes na imagem. Desta maneira é possível alcançar maiores taxas de compressão. Todavia, por razões funcionais e legais, a perda de informações de determinados tipos de imagens deveriam ser evitadas, introduzindo a necessidade de técnicas de compressão sem perdas.

Entre os métodos de compressão de imagens temos a transformada *wavelet*, que faz parte de uma teoria relativamente nova, e tem demonstrado que é uma ferramenta poderosa e vantajosa na área de compressão de imagens. *Wavelets* são funções bases os quais representa uma determinada função em múltiplos níveis de detalhe. *Wavelets* foram desenvolvidas independentemente em diferentes áreas, tais como: matemática, física, engenharia elétrica e geologia. Intercâmbios entre estas áreas têm ocorridos nos últimos quinze anos.

A transformada *wavelet* apresenta algumas vantagens em relação às outras técnicas de compressão de dados. Ela fornece também, uma alternativa para os clássicos métodos de Fourier para análise e síntese de dados em uma e multi-dimensões, e tem numerosas aplicações em diversas áreas tais como: física, astronomia, estatística, sismologia, turbulência, imagens médicas, processamento de imagem, áudio e vídeo, processamento de

sinais, redes neurais, computação gráfica, compressão de dados, transmissão de dados e na solução de equações diferenciais parciais. Por ser uma ferramenta relativamente nova, ainda existem muitas áreas que podem ser exploradas.

Como exemplos da aplicações da transformada *wavelet* em compressão de imagens, podemos citar a compressão de impressões digitais do FBI, que possui um arquivo com mais de 30 milhões de impressões e também o JPEG-2000 [34, 46] que está se tornando um novo padrão internacional para compressão de imagens e que passou a utilizar a transformada *wavelet* em vez da transformada discreta cosseno (DCT).

1.1 Motivação

A investigação, implementação e avaliação de algoritmos para compressão sem perdas, de projeções de tomografia computadorizada, usando a transformada discreta *wavelet* (*discrete wavelet transform* - DWT) formam o objetivo primário desta dissertação.

Vários trabalhos têm sido desenvolvidos na área de compressão de imagens utilizando métodos baseados em *wavelets*. Todavia, foram desenvolvidos poucos trabalhos utilizando a transformada *wavelet* sem perdas. Os trabalhos envolvendo compressão de imagens de tomografia computadorizada geralmente são realizados utilizando imagens já reconstruídas.

Neste trabalho em vez de utilizarmos as imagens já reconstruídas, utilizaremos as projeções de tomografia computadorizada com objetivo de reduzir a quantidade de informação necessária e mantendo a integridade dos dados. Os dados originais também serão preservados, o que permitirá a aplicação de outras técnicas de reconstrução, o qual também é tópico de constante pesquisa. Como estas informações não podem sofrer perdas no processo de compressão e descompressão, deveremos adotar um método de compressão que permita que os dados possam ser reconstruídos para a sua forma original. Para tal, estaremos

utilizando a transformada *wavelet* de inteiros que permite alcançar os melhores resultados em termos de compressão e atendem a este requisito. Métodos que resultam em perdas de informações não são aceitos para este tipo de dados, pois a perda de informações pode prejudicar a reconstrução e consequentemente a análise e o diagnóstico a partir da imagem final.

Nosso interesse nesta investigação abordará apenas as técnicas de compressão de imagens, uma das inúmeras aplicações possíveis com a transformada *wavelet*.

1.2 Histórico

A primeira menção de *wavelets* apareceram no apêndice da tese de Alfred Haar em 1909 [22]. Uma propriedade da *wavelet* de Haar é que ela possui um suporte compacto, ou seja, ela se anula fora de um intervalo finito. As *wavelets* de Haar não são diferenciáveis continuamente, o que limita de certa forma sua aplicação.

A transformada *wavelet* é o resultado do trabalho de um grande número de pesquisadores. Um breve histórico sobre as *wavelets* é apresentado por JAWERTH e SWELDENS [29] e GRAPS [22] do qual faremos um breve resumo a seguir.

Nos anos 30, vários grupos pesquisavam independentemente a representação de funções usando funções de base de escala variável, chamadas funções de base de Haar. O fisico Paul Levy pesquisou movimentos Brownianos, um tipo de sinal randômico, e Littlewood, Paley e Stein descobriram uma função que pode variar em escala e conservar energia quando computando a energia da função.

Ainda nos anos 30, a comunidade matemática percebeu que as técnicas desenvolvidas por Fourier não eram muito adequadas para a solução de muitos problemas e recorreram para às chamadas técnicas de Littlewood-Paley, freqüentemente substitutas eficientes.

Durante os anos 50 e 60, desenvolveram poderosas ferramentas para solucionarem equações diferencias parciais e equações integrais, e perceberam que elas combinavam com a teoria de Calderón-Zygmund, uma área de análise harmônica.

No início dos anos 80, Strömberg descobriu as primeiras *wavelets* ortogonais. Independente dos desenvolvimentos em análise harmônica, Alex Grossmann e Jean Morlet [23] (físico e engenheiro, respectivamente), juntamente com seus colegas de trabalho estudaram a transformada *wavelet* em sua forma contínua. Eles definiram *wavelets* em um contexto de física quântica. Surgia a teoria de "frames".

Em seguida, Yves Meyer, matemático Francês, e seus colaboradores perceberam que as ferramentas da teoria de Calderón-Zygmund, em particular as representações de Littlewood-Paley, poderiam levar a uma concepção unificada de muitos dos resultados em análise harmônica. Começaram também a compreender que aquelas técnicas poderiam substituir as séries de Fourier em aplicações numéricas.

Alex Grossmann e Jean Morlet sugeriram a palavra *wavelet* para os blocos construtivos, e o que antes se chamava teoria de Littlewood-Paley, passou a ser chamado teoria *wavelet.*

Pierre-Gilles Lemarié e Yves Meyer, independente de Strömberg, construíram novas expansões de *wavelet* ortogonal. Em 1985, Stephane G. Mallat [32] introduziu o conceito de análise em multiresolução, dando um novo impulso a essa teoria. Ele descobriu algumas relações entre filtros de quadratura espelhada (*quadrature mirror filters* – QMF), algoritmos piramidais e bases de *wavelets* ortonormais. Inspirado em parte desses resultados, Yves Meyer construiu a primeira *wavelet* não trivial. Ao contrário da *wavelet* de Haar, as *wavelets* de Meyer são diferenciáveis continuamente; todavia elas não têm suporte compacto. A introdução da análise em multiresolução e a transformada *wavelet* rápida por Mallat e

Meyer forneceu a conexão entre filtros sub-bandas e *wavelets*. Alguns anos depois, em 1988, usando o trabalho de Mallat, Ingrid Daubechies [11] construiu uma família de *wavelets* com suporte compacto.

Desde então, muitos trabalhos têm sido desenvolvidos em diferentes áreas, sendo uma delas, as aplicações em compressão de imagens digitais.

1.3 Trabalhos Relacionados

Nesta seção, apresentaremos alguns dos trabalhos mais recentes na área de compressão de imagens baseados na transformada *wavelet*. Os algoritmos de compressão de imagens propostos aqui são ambos com perdas e sem perdas.

Entre os trabalhos mais importantes envolvendo a transformada *wavelet*, temos os trabalhos desenvolvidos por Antonini *at al.* [1], DeVore *at al.* [15], Lewis e Knowles [31], Shapiro [55], Said e Pearlman [33, 50, 51, 52] e Zandi *at al.* [53, 72]. Temos ainda, os algoritmos da segunda geração de *wavelets* que mapeiam inteiros para inteiros [4, 5] utilizando os "*lifting schemes*" propostos por Sweldens [12, 59, 60, 61, 62, 64, 65].

No trabalho de Antonini *at al.* [1], é apresentado um método de compressão que associa a transformada *wavelet* e quantização vetorial. Os autores decompõem a imagem original em diferentes escalas, usando o algoritmo em pirâmide, explorando as características psicovisuais nos domínios do espaço e da freqüência. Em seguida, é aplicado a quantização vetorial e um esquema de alocação de bits para os coeficientes *wavelets*. Neste artigo, é apresentado também um esquema de transmissão progressiva.

DeVore *at al.* [15] apresenta uma nova teoria matemática para análise de métodos de compressão de imagens baseado na decomposição *wavelet*. Esta teoria precisamente relata (a) a taxa de decaimento no erro entre a imagem original e a imagem comprimida quando o tamanho da imagem comprimida aumenta (ou seja, quando a quantidade de compressão

diminui) para (b) a suavidade da imagem em certas classes de suavidade chamado espaços Besov ($B_q^\alpha(L^q(I))$). Esta teoria limita o erro causado pela quantização dos coeficientes *wavelets*. Baseado em pesquisas experimentais anteriores na resposta amplitude-freqüência-espacial do sistema visual humano (*human visual system* – HVS), eles discutem também que em muitas instâncias a medida do erro, causado na compressão de imagem, utilizando o erro médio quadrático (*mean square error* – MSE) (no espaço L^2) pode não ser sempre apropriado e que o erro médio absoluto (*mean absolute error*) (no espaço L^1) é mais apropriado em aplicações de compressão de imagens.

O conceito de *zerotrees* foi originalmente introduzido por Lewis e Knowles [31]. Eles propuseram um esquema de compressão de imagens baseado na decomposição dos componentes espectrais e espaciais da imagem usando *wavelets* ortogonais bidimensional, para em seguida aplicar um algoritmo de compressão que explore as características do sistema visual humano. Os coeficientes são codificados hierarquicamente e quantizados individualmente.

Shapiro [55] desenvolveu um esquema de codificação de imagem usando uma estrutura hierárquica de "*zerotrees*" dos coeficientes *wavelets*. Este algoritmo é chamado EZW (*Embedded Zerotree Wavelet*) e possui similaridades com o algoritmo posposto por Lewis e Knowles [31]. O *zerotree* é um método eficiente de codificação embutida (*embedded*) dos coeficientes *wavelets*, os quais toma vantagem da similaridade natural das diferentes bandas no domínio da transformada.

O algoritmo EZW é baseado em quatro conceitos: (1) transformada discreta *wavelet* ou decomposição hierárquica em sub-bandas, (2) predição da ausência de informação significante através das escalas explorando a auto-similaridade natural em imagens, (3)

quantização por aproximação sucessiva codificada por entropia e (4) compressão sem perdas os quais é realizado via codificação aritmética adaptativa.

Para um dado limiar inicial *T*, um coeficiente *wavelet x* é dito ser significante com respeito a *T* se | *x* | >= *T*. Um coeficiente *wavelet* é dito ser um elemento *zerotree* raiz com respeito ao limiar *T* se ele e todos os seus descendentes (figura 1.1) são insignificantes com respeito a *T*. *Zerotrees* são construídas para o limiar *T*, e os símbolos descrevendo a estrutura *zerotree* são codificados através da codificação aritmética adaptativa. O limiar é reduzido pela metade e o processo repetido. A codificação pode finalizar em qualquer estágio. Uma explicação mais detalhada e um exemplo simples deste esquema pode ser obtido em [55].

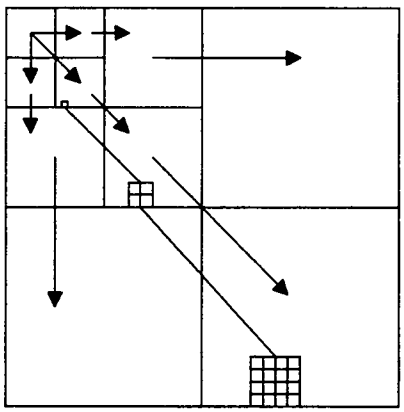


Figura 1.1 – Dependências pai-filho entre os coeficientes *wavelet* em diferentes sub-bandas.

O algoritmo EZW produz resultados de compressão que são competitivos com todos os algoritmos de compressão conhecidos em imagens de teste padrão. O EZW sugere também uma maneira eficiente para ordenação dos bits dos coeficientes *wavelet* para transmissão.

Mais recentemente, um outro algoritmo chamado SPIHT (*Set Partitioning in Hierarchical Trees*) foi proposto por Said e Pearlman [50], os quais é uma implementação mais eficiente do esquema de codificação *zerotree* do Shapiro [55]. Este algoritmo permite codificar uma imagem com perdas ou sem perdas. Ele usa um esquema de multiresolução em

pirâmide ao quais é melhorado via codificação preditiva através da transformada S+P (transformada Seqüencial + Predição) [4, 51, 52]. Ele difere de outros métodos no fato de que a predição é usada durante (em vez de depois) a seqüência de transformações recursivas, e consequentemente pode usar informações que não estão disponíveis depois que a imagem é transformada. O algoritmo SPIHT é baseado em três estágios: (*i*) aplicação da tranformada S+P para decompor a imagem, (*ii*) uma técnica de codificação baseado em *zerotree* e (*iii*) codificação aritmética adaptativa dos dados residuais.

Esta técnica produz significantemente melhor compressão do que os métodos de compressão *wavelet* tradicional com similar complexidade computacional, e representa o estado da arte em compressão de imagens de propósito geral [33].

A transformada S (Seqüencial) utilizada nesta técnica é uma adaptação da transformada de Haar para mapear os valores de inteiros para inteiros, permitindo assim a reconstrução exata das informações. No capítulo 4, descreveremos mais a respeito deste algoritmo.

No trabalho de Zandi *at al.* [53, 72] é proposto um esquema de compressão de imagens reversível (exata reconstrução). CREW (*Compression with Reversible Embedded Wavelets*) é um sistema de compressão de imagens de tons contínuos unificado, ou seja, permite a compressão tanto com perdas quanto sem perdas. CREW também é baseado na transformada *wavelet* de inteiros, sendo utilizado a transformada TS (*Two-Six Transform*). Ele é baseado na transformada *wavelet* usando uma aproximação reversível de um dos melhores filtros *wavelet* conhecido e sua performance é igual ou melhor do que outros métodos existentes. Para codificação dos coeficientes, CREW usa um método similar ao *zerotree* do Shapiro [55], e um método chamado *Horizon*. Codificação *Horizon* é uma codificação baseada em contexto que toma vantagem da informação espacial e espectral

disponível no domínio *wavelet*. Os coeficientes são codificados de modo a permitir a compressão com perdas por simplesmente truncando os dados após a aplicação da transformada. Três codificadores de entropia foram experimentados: *finite state machine coder*, *parallel coder* e *Q-coder*. CREW fornece o estado da arte para compressão sem perdas de imagens médicas (acima de 8 bits por *pixel*), e compressão com perdas e sem perdas de imagens com 8 bits. Este esquema tem também uma implementação simples tanto em software quanto em hardware. CREW foi submetido ao comitê ISO/IEC JTC1/SC29/WG1 (formalmente os comitês JPEG e JBIG) como um padrão internacional.

Por fim, Calderbank *at al.* [4] apresenta duas abordagens para construir transformadas *wavelet* de inteiros para inteiros os quais podem ser usadas para compressão sem perdas. Na primeira parte do trabalho é apresentado uma adaptação do pré codificador desenvolvido por Laroia, Tretter e Farvardin. O leitor pode consultar [4] para obter mais detalhes. Na segunda abordagem é apresentado como conseguir uma implementação reversível usando o *lifting scheme* proposto por Sweldens [12, 59, 60, 61, 62, 64, 65]. Esta segunda abordagem é discutida em mais detalhes no capítulo 4.

1.4 Estrutura da Dissertação

A dissertação está estruturada em seis capítulos. No segundo capítulo, apresentamos duas maneiras de classificar as técnicas de compressão de imagens: as técnicas de compressão de imagens com perdas e sem perdas, incluindo também as técnicas de compressão por transformada.

No terceiro capítulo descrevemos inicialmente a transformada de Fourier e comparamos com a transformada *wavelet*. Em seguida apresentamos a transformada *wavelet* direta e inversa, começando com uma descrição da transformada *wavelet* contínua e mais adiante introduzimos o conceito de análise em multiresolução e a sua relação com banco de

filtros. Além disso, é apresentado também como a transformada *wavelet* pode ser usada em aplicações de compressão de imagens, partindo inicialmente de uma análise unidimensional, que torna mais simples a sua compreensão, e posteriormente mostrando sua aplicação em sinais bidimensionais.

O quarto capítulo aborda a transformada *wavelet* que mapeia inteiros para inteiros, o qual permite a reconstrução exata dos dados de entrada. Inicialmente, apresentamos as transformadas S, S+P e TS, as quais são baseadas na transformada *wavelet* de inteiros. Em seguida, descrevemos o *lifting scheme* e como ele pode ser mapeado para inteiros. Nesta seção, apresentamos também as transformadas *wavelet* de inteiros reversíveis utilizadas nos experimentos deste trabalho.

No quinto capítulo, descrevemos o tipo de dados que estaremos utilizando na realização dos experimentos e uma descrição do processo de compressão adotado, o qual foi baseado no *lifting scheme* com a propriedade de preservação de precisão [6]. Este capítulo, apresenta também os resultados obtidos na compressão das projeções. Por fim, concluímos o trabalho e apresentamos propostas para trabalhos futuros.

2. Compressão de Imagens

A compressão de uma imagem é possível porque as imagens, em geral, apresentam um alto grau de coerência, que se traduz em uma redundância de informação quando codificada [20]. Por exemplo, tomando um *pixel* (*picture element*) qualquer de uma imagem, provavelmente, a cor desse *pixel* será igual a dos elementos vizinhos ou de uma outra região próxima na imagem, porque há uma grande probabilidade de todos eles pertencerem a um mesmo objeto da imagem. Caso isso não ocorra, certamente uma relação mais complexa existirá na imagem. Baseado neste fato, os métodos de compressão de imagens digitais visam produzir, através da redução ou eliminação de redundância, um código mais compacto que preserve as informações contidas na imagem. A redundância é uma característica que está relacionada a distribuição estatística da informação presente na imagem.

Em uma imagem digital há, basicamente, três tipos de redundância de dados que pode ser identificado e explorado [21]: redundância de código, redundância interpixel e redundância psicovisual.

Para reduzir a redundância de código, o processo de codificação determina um código binário com tamanho variável (*variable-length coding* – VLC). Aos valores que ocorrem mais freqüentemente na imagem, é atribuído um código menor (em número de bits), enquanto que os valores que são menos freqüentes, um código maior é atribuído.

A redundância interpixel permite prever razoavelmente o valor de um *pixel* a partir do valor do *pixel* de seus vizinhos, com isso a informação dos *pixels* a ser codificada é relativamente pequena. Esta correlação espacial resulta da relação estrutural ou geométrica entre os objetos na imagem. Uma variedade de nomes têm sido criados para referenciar para essas dependências interpixels, tais como: redundância espacial, redundância geométrica e

redundância *interframe* (ou temporal, que ocorre entre *frames* adjacentes em uma seqüência de imagens). Podemos relacionar ainda a redundância espectral que ocorre entre diferentes planos de cores ou bandas espectrais.

A redundância psicovisual ocorre pelo fato de algumas informações terem menos importância do que outras informações em um processamento visual normal. Essas redundâncias podem ser eliminadas sem comprometer a qualidade da imagem. Todavia, a eliminação de redundância psicovisual resultará numa perda de informação, o que leva a um processo de compressão com perdas.

Dependendo da área de aplicação, as informações que desejamos preservar podem ser de natureza objetiva ou subjetiva [20]. No primeiro caso o método de compressão deve permitir a recuperação exata dos dados da imagem original. Dizemos, nesse caso, que o processo é reversível ou que temos uma compressão sem perda (*lossless*). Caso contrário, ele é chamado irreversível, e dizemos que temos uma compressão com perda (*lossy*), isto é, ele não permite a reconstrução exata dos dados originais.

As técnicas ditas reversíveis são também chamadas de redução de redundância ou sem ruído, enquanto que as técnicas irreversíveis são conhecidas também por redução de entropia ou com ruído [35].

Os recentes avanços das técnicas de compressão com perdas incluem diferentes métodos tais como a transformada discreta cosseno (*discrete cosine transform* - DCT), quantização vetorial, transformada *wavelet*, redes neurais, codificação fractal e outras técnicas por transformada. Apesar destes métodos permitirem a obtenção de altas taxas de compressão (tipicamente 50:1 ou maior, dependendo da qualidade desejada), eles não permitem a reconstrução exata da versão original dos dados de entrada. Portanto, métodos com perdas não são aceitáveis para determinadas aplicações, tais como: imagens médicas

digitais, imagens de satélite, dados sísmicos, imagens de alta fidelidade, arquivos executáveis, e assim por diante. Técnicas de compressão sem perdas, todavia, permitem a reconstrução exata dos dados originais, mas as taxas de compressão são comparativamente baixas (cerca de 2:1 a 4:1).

Diferentes fórmulas são utilizadas para o cálculo das taxas de compressão. Algumas pessoas preferem calcular em bits/bytes. Esta medida de compressão fornece o número de bits necessários por *pixel* depois da compressão. O valor bits-por-*pixel* (bpp) permite comparar os métodos de compressão e permite calcular o tamanho da imagem comprimida e a taxa de transmissão necessária para qualquer tamanho de imagem. Outros preferem utilizar razões, por exemplo: 2:1. Isto significa que a quantidade de dados foi reduzida pela metade. Usaremos neste trabalho a fórmula (em porcentagem) definida por:

$$\left(1-\frac{dados\ comprimidos}{dados\ originais}\right)*100 \qquad (2.1)$$

Isto significa que um arquivo que não muda de tamanho após a compressão terá uma taxa de compressão de 0% (zero por cento). Um arquivo comprimido pela metade do seu tamanho original terá uma taxa de compressão de 50%. Teoricamente a taxa de compressão máxima é 100%. Isso só ocorrerá se o tamanho do arquivo comprimido for igual a zero, os quais ocorre somente se o tamanho dos dados originais é zero também. Se o tamanho do arquivo comprimido for maior do que o arquivo original teremos uma taxa de compressão negativa.

2.1 Compressão Sem Perdas

Compressão de dados sem perdas é o método de compressão que permite a recuperação exata dos dados originais após o processo de descompressão. Os esquemas de compressão de imagens sem perdas freqüentemente consiste de dois componentes distintos e

independentes: modelagem e codificação.

Em uma quantização uniforme o número de bits utilizado para codificar cada nível é chamado de taxa de bits. Se a codificação tem L níveis então a taxa de bits é portanto $b = \log_2 L$. Se a codificação não for uniforme, o número b acima fornece apenas o número médio de bits utilizados para codificar cada nível de quantização [20]. Nesse caso ele é chamado de taxa média de bits.

Para decidir se uma determinada taxa média de bits é adequada para codificar uma imagem, precisamos ter uma medida da "quantidade de informação" presente na imagem. Essa medida é dada pelo cálculo da entropia da imagem.

A entropia, termo usado na teoria da informação, indica o número médio de bits necessários para codificação não ambígua de cada símbolo de um alfabeto fonte. Todos os algoritmos sem perdas requererão igual ou mais bits para a compressão. Uma breve introdução da teoria da informação será apresentada a seguir.

2.1.1 Teoria da Informação

De acordo com Claude Shannon [54], a entropia é a compressão máxima possível quando nós codificamos dentro de um mesmo alfabeto. Se S é uma variável aleatória com L valores (v_1, v_2, ..., v_L) com suas respectivas probabilidades (P_1, P_2, ..., P_L), então a entropia de S, que indicamos por H, pode ser definida como [20, 21]:

$$H = -\sum_{i=1}^{L} P_i \log_2 P_i = \sum_{i=1}^{L} P_i \log_2 \frac{1}{P_i}, \qquad (2.2)$$

onde P_i é a probabilidade de ocorrência de um nível de quantização na codificação da imagem, e $\log_2 \frac{1}{P_i}$ indica o número de bits necessários para codificar cada nível de quantização.

Logicamente, o conjunto de probabilidades deve satisfazer à condição

$$\sum_{i=1}^{L} P_i = 1. \qquad (2.3)$$

Claramente podemos observar que $P_i \geq \frac{1}{L}$, logo

$$\log_2 \frac{1}{P_i} \leq \log_2(L). \qquad (2.4)$$

Usando a desigualdade acima, juntamente com a equação 2.2 , temos:

$$H \leq \sum_{i=1}^{L} P_i \log_2(L) = \log_2(L) \sum_{i=1}^{L} P_i = \log_2(L). \qquad (2.5)$$

Ou seja, a entropia H satisfaz à desigualdade

$$0 \leq H \leq \log_2(L). \qquad (2.6)$$

A teoria de Shannon determina que se o tamanho dos dados é n, então $nH(S)$ é o número de bits requerido para representar S e é então chamado a *complexidade* de S.

O cálculo da entropia da equação 2.2 , também chamado de estimativa de primeira ordem (*first order estimate*) [21], fornece uma estimativa que é o menor limite, para a taxa média de bits, que pode ser realizado através da codificação com tamanho variável somente, sem haver perdas na compressão dos dados. A codificação de tamanho variável é usada para reduzir redundâncias de código. A diferença entre a estimativa da entropia de mais alta ordem e a estimativa de primeira ordem indica a presença de redundâncias interpixel.

Desse modo, o conhecimento da entropia de uma imagem, nos fornece um parâmetro importante para que possamos verificar a qualidade da taxa média de bits obtida em uma determinada codificação. Aplicando métodos de compressão com perdas pode-se produzir uma taxa média de bits inferior à entropia da imagem.

Os algoritmos de compressão sem perdas podem ser divididos, basicamente, em duas categorias: os métodos baseados em estatística e os métodos baseados em dicionário. Entre estes métodos os mais conhecidos são: a codificação Shannon-Fano [25, 39, 54], a codificação de Huffman [21, 25, 39], a codificação aritmética [21, 36, 39, 70], os algoritmos baseados na codificação Lempel-Ziv (LZ) [39, 69, 73, 74] e a codificação *Run-Length Encoding* (RLE) [21]. Existem também os algoritmos, não baseados em *wavelets*, desenvolvidos para explorar a redundância interpixel das imagens tais como: Modulação por Código de Pulso Diferencial (*Differential Pulse Code Modulation* – DPCM) [28], LJPEG (*Lossless Joint Photographic Experts Group*) [13, 14, 67], JBIG (*Joint Bi-level Imaging Group*), BTPC (*Binary Tree Predictive Coding*) [13, 14], CALIC (*Context-based Adaptive Lossless Image Codec*) [71], FELICS (*Fast, Efficient, Lossless Image Compression System*) [27], LOCO-I (*LOw COmplexity LOssless COmpression for Images*) [68], entre outros.

2.2 Compressão com Perdas

Até aproximadamente 1980, a maioria das técnicas de codificação de imagens eram baseados na clássica teoria da informação para explorar as redundâncias estatísticas presentes nas imagens com objetivo de realizar compressão [48]. As técnicas usadas eram baseadas em *pixels* e não faziam nenhum uso das informações contidas na imagem. Removendo apenas estas redundâncias nas imagens, podemos obter somente uma taxa de compressão limitada. Se aumentar a taxa de compressão resultará na remoção de dados não redundantes e produzirá uma degradação visual das imagens. Para obter taxas de compressão mais elevadas e manter as imagens reconstruídas com uma qualidade aceitável, outras técnicas de compressão devem ser consideradas.

Foram desenvolvidas novas técnicas de compressão que apresentam melhor desempenho do que as técnicas de codificação de imagens da primeira geração. Estes

métodos tentam identificar características dentro da imagem e usar estas características para realizar compressão. Estes recentes desenvolvimentos são chamados de segunda geração de codificação de imagens [48]. Todas as técnicas da segunda geração incorporam propriedades do sistema visual humano (*human visual system* – HVS) como estratégia de codificação com o objetivo de realizar altas taxas de compressão enquanto mantendo uma qualidade aceitável da imagem.

Os algoritmos de compressão de imagens com perdas procuram tirar vantagem das limitações da visão humana. Com o objetivo de aumentar a taxa de compressão, algumas informações podem ser eliminadas, sendo que a perda de algumas informações podem não ser percebidas decorrente do fato da visão humana ser mais perceptível a luminosidade do que a variação de cores, todavia, a perda de informações pode provocar o surgimento de artefatos na imagem comprimida e consequentemente deterioração da qualidade. Na sua maioria, as técnicas de compressão com perdas se baseiam em três etapas: decomposição ou transformação da imagem, estratégias de quantização/limiar e técnicas de modelagem e codificação sem perdas. Este processo de compressão é ilustrado no diagrama abaixo.

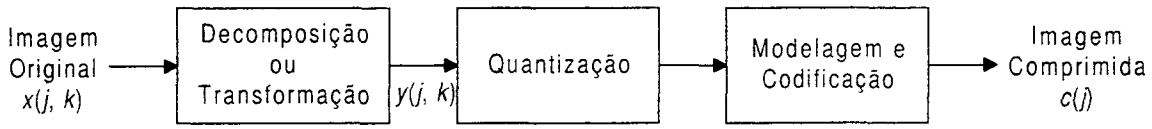


Figura 2.1 – Processo de compressão com perdas.

Partindo de uma imagem original $x(j, k)$, a decomposição ou transformação descorrelaciona os dados da imagem, mapeando os dados para um novo domínio $y(j, k)$.

Na etapa de quantização, duas técnicas diferentes de quantização podem ser aplicadas: a quantização escalar ou a quantização vetorial. Na quantização escalar, os dados de entrada são quantificado isoladamente. A quantização vetorial é uma generalização da quantização escalar em que vetores, ou blocos, de *pixels* são quantificados em vez dos *pixels*

separadamente. Geralmente nessa etapa, costuma-se aplicar um limiar com o objetivo de diminuir o número de coeficientes diferentes de zero, com isso aumenta-se a taxa de compressão. Em razão disso, esta etapa geralmente implica na perda de informações, impedindo a reconstrução exata dos dados. Quando não ocorrer nenhuma quantização/limiar, é possível recuperar completamente os dados originais. Desde que a transformação seja reversível.

Por fim, a etapa de codificação tem a função de transformar os dados para uma representação unidimensional $c(j)$, podendo ser utilizado uma das técnicas de codificação de entropia, tais como: codificação de Huffman, codificação Shannon-Fano ou codificação aritmética. A reconstrução da imagem é realizada efetuando as operações inversas do processo de compressão.

Se as operações de transformação direta e inversa pudessem ser computadas com exatidão, poderíamos obter a reconstrução exata do sinal original. Muitas das técnicas de compressão com perdas poderiam ser adaptadas para se tornarem sem perdas desde que não seja introduzido um erro na etapa de transformação. Mas, normalmente é a introdução de um estágio de erro no processo que fornece a facilidade para a compressão [48].

Quando se trata de compressão com perdas, a medida da taxa de compressão não é tão significativa se não houver meios de se comparar a qualidade da imagem. Como tais comparações são subjetiva, recorremos freqüentemente a medidas quantitativas. A distorção (inversa da qualidade da imagem) é uma medida da distância entre a imagem original e a imagem reconstruída. Existem várias maneiras de calcular a distorção, entre as mais usadas temos o erro médio quadrático (*mean square error* – MSE) e a relação sinal de ruído de pico (*peak signal noise ratio* – PSNR). A clássica teoria da taxa de distorção (*rate-distortion* – R-D) preocupa-se com a tarefa de representar os dados com o menor número de bits possível

para uma determinada qualidade [43].

Para uma imagem de tamanho M x N, sendo $x(m, n)$ a imagem original e $\hat{x}(m, n)$ a imagem reconstruída a partir dos dados comprimidos, o cálculo do erro médio quadrático é dado por

$$MSE = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left|x(m,n)-\hat{x}(m,n)\right|^2 \qquad (2.7)$$

e a relação sinal de ruído de pico é dado por

$$PSNR_{[dB]} = 10\log_{10}\left(\frac{L^2}{MSE}\right) \qquad (2.8)$$

sendo L o valor pico-a-pico do dado da imagem original. Para uma imagem com 8 bits por *pixel*, por exemplo, este valor é igual a 255.

2.2.1 Codificação por Transformada

As transformações matemáticas são aplicadas às imagens com o intuito de descorrelacionar os dados. Existe uma grande variedade de métodos de compressão baseados em transformadas matemáticas, que são empregadas no mapeamento de um domínio para outro, entre elas podemos citar: transformada de Fourier, Cosseno, Seno, Karhunen-Loève, Walsh, Hadamard, Haar, entre outras. Estas transformações são ortogonais e unitárias garantindo a transformação inversa. Mais recentemente se incorporam a essa relação a transformada *wavelet* e a transformada fractal.

O termo codificação por transformada genericamente descreve as técnicas de codificação onde os dados originais são primeiro decompostos usando uma transformação linear e em seguida cada um dos componentes de freqüência obtidos é quantizado e codificado.

Na maioria das imagens, após a transformação, uma grande quantidade de coeficientes tornam-se pequenos em magnitude, assim eles podem ser grosseiramente quantizados ou completamente rejeitados sem prejudicar a qualidade da imagem reconstruída. Com a descorrelação dos dados, é possível obter também uma representação mais compacta na etapa de codificação.

2.2.2 Padrões de Compressão de Imagens

Dois padrões da ISO (*International Standards Organization*) bastante conhecidos para compressão de imagens, baseados em transformada, são os esquemas JPEG (*Joint Photographic Experts Group*) [21, 39, 67] para imagens estáticas e o MPEG (*Moving Picture Experts Group*) para vídeo. Os dois esquemas são baseados na transformada discreta cosseno. Temos ainda o LJPG [13, 14, 67] e o JPEG-LS [34, 68] para compressão sem perdas, o JBIG para compressão sem perdas de imagens binárias e o JPEG-2000 [34, 46], um novo padrão que está surgindo para compressão de imagens com perdas ou sem perdas baseado na transformada *wavelet*.

A necessidade de um padrão internacional para compressão de imagens estáticas resultou, em 1986, na formação do JPEG. Formalmente conhecido como ISO/IEC JTC1/SC29/WG1. O objetivo deste grupo foi desenvolver um método para compressão de imagens de tons contínuos que atendesse aos seguintes requisitos:

1. A qualidade da imagem reconstruída deve ser a mais perfeita possível quando comparada com a original. O codificador deve ser parametrizado permitindo a escolha da taxa de compressão e qualidade da imagem desejável.
2. Ser aplicável para praticamente qualquer tipo imagem digital de tons contínuos e não devem haver restrições com relação ao conteúdo da cena, tais como complexidade, número de cores ou propriedades estatísticas.

3. Ter uma complexidade computacional razoável, permitindo sua implementação tanto em software quanto em hardware.
4. Ter os seguintes modos de operação:
   - Codificação seqüencial. Cada componente da imagem é codificado com uma simples varredura da esquerda para a direita e de cima para baixo.
   - Codificação progressiva. A imagem é codificada em múltiplas varreduras para aplicações no qual o tempo de transmissão é longo.
   - Codificação sem perdas. A imagem é codificada garantindo a reconstrução exata da imagem original.
   - Codificação hierárquica. A imagem é codificada em múltiplas resoluções.

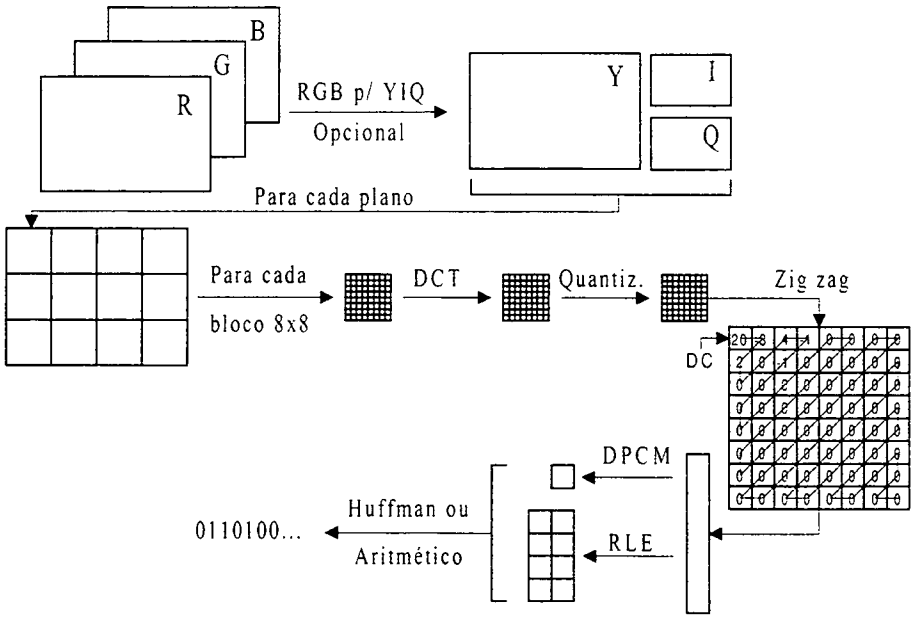


Figura 2.2 – Esquema de codificação do JPEG.

Inicialmente, cada componente de cor da imagem é dividida em blocos não sobrepostos de 8 × 8 *pixels* e então trata cada um destes blocos de forma independente durante a compressão. Em seguida, é aplicada a transformada discreta cosseno direta (*Forward DCT* - FDCT) bidimensional. O componente do canto superior esquerdo é

chamado de componente DC do bloco e representa o valor médio do bloco. Os outros 63 coeficientes são denotados coeficientes AC. Utilizando a DCT observamos uma concentração da energia do bloco no coeficiente DC e nos coeficientes AC as baixas freqüências. Os coeficientes DC são codificados usando DPCM. São utilizados os coeficientes DC do bloco atual e do bloco anterior. Os 63 coeficientes AC são quantizados uniformemente usando uma tabela específica. O CCITT define 4 tabelas de quantização e a tabela deve ser escolhida de acordo com as características do bloco. Os 63 coeficientes AC são ordenados em forma de *zig-zag* da menor até a freqüência mais elevada (conforme mostrado na figura 2.2). Após a etapa de quantização, os blocos apresentam um elevado número de zeros. A seguir, esses coeficientes são codificados utilizando RLE. Na etapa final, os coeficientes são codificados usando um código de tamanho variável (VLC). Dois métodos podem ser utilizados: codificação de Huffman ou codificação aritmética. A transformada inversa consiste em aplicar as etapas em ordem reversa.

Como visto anteriormente, a recomendação JPEG especifica também um método de compressão sem perdas. A compressão sem perdas utiliza técnicas de predição ao invés da DCT.

Uma expectativa comum com relação ao uso da transformada *wavelet* é que ela produz uma melhor qualidade subjetiva da imagem do que o codificador JPEG padrão. Em vista disso, o JPEG está propondo um novo padrão para compressão de imagens chamado JPEG-2000 [34, 46]. Esse novo paradigma de compressão de imagens utilizará a transformada *wavelet* ao invés da DCT.

O projeto do JPEG-2000 foi motivado com a submissão do algoritmo CREW [53, 72] (resumo descrito na seção 1.3) para a padronização de compressão sem perdas (*lossless*) e quase sem perdas (*near-lossless*) para imagens de tons contínuos, conhecido atualmente

como JPEG-LS. Embora o LOCO-I [68] tenha sido selecionado como a base para o JPEG-LS. Todavia, foi reconhecido que o CREW possui muitas características que mereciam o desenvolvimento de um novo padrão [34]. Em 1996, o JPEG-2000 foi aprovado como um novo item de trabalho. Muitas das idéias no JPEG-2000 foram inspiradas nos trabalhos de [50, 55, 72].

JPEG 2000 refere-se para todas as partes do padrão. A proposta atual é constituída de 6 partes. As partes são: [34]

- Parte 1, Sistema de codificação de imagens JPEG 2000 (o núcleo)
- Parte 2, Extensões (adicionar mais características e sofisticação para o núcleo)
- Parte 3, *Motion* JPEG 2000
- Parte 4, Adaptação
- Parte 5, Software de referência (atualmente implementações em Java e C)
- Parte 6, Formato de arquivo para imagens compostas.

O padrão JPEG-2000 terá como característica a eficiência da compressão melhorada (estimada em 30% dependendo do tamanho e da taxa de bits da imagem), permitirá compressão tanto com perdas quanto sem perdas, múltiplas resoluções, decodificação progressiva, codificação de região de interesse (*Region Of Interest* – ROI), o qual permite selecionar partes de uma imagem para ser codificada com mais alta qualidade, e ainda, tudo em um único fluxo de bits comprimido. Permitindo as aplicações manipularem ou transmitirem somente as informações essenciais para um dispositivo a partir da imagem fonte comprimida [34, 46].

O JPEG-2000 tem muitas características novas em relação ao JPEG padrão, algumas delas são:

- A performance em compressão constitui o estado da arte para baixa taxa de bits;
- Transmissão progressiva por qualidade, resolução, componentes ou localidade espacial;
- Compressão com perdas e sem perdas unificado;
- Acesso aleatório para fluxo de bits (*bitstream*);
- Processamento no domínio comprimido (por exemplo, rotação e corte);
- Codificação da região de interesse por progressão (permite transmitir a região de interesse primeiro durante a transmissão progressiva).

A complexidade da transformada *wavelet* no JPEG-2000, depende do tamanho do filtro e dos filtros de ponto flutuante ((9,7), (10,18)) *versus* filtros inteiros ((13, 7), (3, 5), (5, 3), (2,10), etc.) utilizados. A Parte I exigirá uma *wavelet* de ponto flutuante (9, 7) e uma inteira (3, 5), enquanto a Parte II permitirá *wavelets* múltiplas incluindo as definidas pelo usuário [34].

A transformada *wavelet* diádica (piramidal) com L-níveis de decomposição é realizada usando a *wavelet* de ponto flutuante (9, 7) ou a *wavelet* de inteiros (5, 3) [34]. Depois da transformação, uma quantização escalar uniforme é empregada em todos os coeficientes *wavelet*. Uma tabela de quantização (*Q-table*), análoga ao JPEG padrão é utilizada. Quando a transformada *wavelet* de inteiros é empregada, o tamanho do passo da quantização escalar é igual a 1, isto é, sem quantização, os quais permitiu progressão sem perdas da mesma maneira do CREW ou SPIHT. Codificação de entropia (*entropy coding*) é realizado independentemente em cada bloco de código (*code-block*). Toda a codificação é feita usando o codificador aritmético binário dependente do contexto. O *MQ-coder* foi aceito como o codificador aritmético para o JPEG-2000.

Um anexo do padrão JPEG-2000 contém um formato de arquivo opcional para incluir informações tais como o espaço das cores dos *pixels* e informações de propriedade intelectual (direitos autorais) para a imagem. Este opcional formato de arquivo é extensível e a parte II definirá armazenamento de muitos tipos adicionais de metadados. A extensão utilizada para representar o formato do arquivo é ".JP2".

3. Transformada Wavelet

3.1 Introdução

As transformações matemáticas são empregadas no mapeamento de funções de um domínio para outro. As transformações são particularmente importantes em processamento e análise de sinais porque no domínio transformado algumas propriedades relevantes do sinal ficam mais evidentes. Para a compressão de imagens, um dos métodos mais utilizados é a codificação por transformada, cujo objetivo principal da transformada é produzir um conjunto de valores representando os *pixels* reordenados, evidenciando a maior concentração de energia possível em menor número de coeficientes.

Como já vimos, no capítulo anterior, existe vários métodos de transformações que podem ser aplicados a um sinal, entre os quais a transformada de Fourier é a mais popular. A transformada de Fourier utiliza funções bases (senos e cossenos) para analisar e reconstruir um sinal, além disso, elas são funções ortogonais, as quais possuem propriedades desejáveis para a sua reconstrução. Tanto a transformada de Fourier quanto a transformada *wavelet* são ambas reversíveis, porém, a transformada de Fourier possui algumas limitações. Como veremos a seguir a transformada *wavelet* possui algumas vantagens, em especial nos casos onde a transformada de Fourier não apresenta bons resultados.

3.2 A Transformada de Fourier

A transformada de Fourier foi descoberta no início do século XIX, pelo matemático francês Joseph Fourier, que mostrou que qualquer função periódica pode ser representada como uma soma infinita de funções exponenciais complexas periódicas. Atualmente, a transformada de Fourier tem sido utilizada em inúmeras aplicações em

processamento de sinais. Na transformada de Fourier uma função no domínio do tempo é mapeada em uma função no domínio da freqüência, onde o seu conteúdo pode ser analisado. Esta transposição ocorre porque a transformada de Fourier expande a função original em termos de funções senos e cossenos de duração infinita. A transformação inversa de Fourier transforma o sinal do domínio da freqüência para o domínio do tempo.

Seja $x(t)$ uma função contínua da variável real t. A transformada de Fourier de $x(t)$, denotado por $\Im\{x(t)\}$, é definida pela equação [21]

$$\Im\{x(t)\} = X(f) = \int_{-\infty}^{+\infty} x(t) \cdot e^{-j2\pi f t}\, dt \qquad (3.1)$$

ou seja, um produto interno do sinal $x(t)$ com um conjunto de exponenciais complexas, que constituem uma base ortonormal. Na equação acima, t é usado para representar o tempo, f a freqüência e j é igual a $\sqrt{-1}$.

Dado $X(f)$, $x(t)$ pode ser obtido usando a transformada inversa de Fourier (*inverse Fourier transform* – IFT) definida pela equação:

$$\Im^{-1}\{X(f)\} = x(t) = \int_{-\infty}^{+\infty} X(f) \cdot e^{j2\pi f t}\, df \qquad (3.2)$$

As equações 3.1 e 3.2 existem se $x(t)$ é contínua e integrável e $X(f)$ é integrável. Na transformada de Fourier o X denota o sinal no domínio do tempo e na transformada inversa de Fourier o x denota o sinal no domínio da freqüência.

A transformada de Fourier pode ser facilmente estendida para uma função de duas varáveis $f(x, y)$. Se $f(x, y)$ é continua e integrável e $F(u, v)$ é integrável, a transformada de Fourier é definida pela equação

$$\Im\{f(x,y)\} = F(u,v) = \int_{-\infty}^{+\infty}\int f(x,y) \cdot e^{-j2\pi(ux+vy)}\, dx\, dy \qquad (3.3)$$

e a transformada inversa de Fourier é definida por:

$$\Im^{-1}\{F(u, v)\} = f(x, y) = \int_{-\infty}^{+\infty}\int F(u, v) \cdot e^{j2\pi(ux+vy)}\, du\, dv \tag{3.4}$$

onde u e v são as variáveis de freqüência.

Para uma seqüência de duração finita, é possível desenvolver uma representação alternativa da transformada de Fourier, referida como transformada discreta de Fourier (*discrete Fourier transform* – DFT). Uma vez que a transformada de Fourier não pode ser computada, a DFT, que é uma versão amostrada, permite o seu cálculo computacional, e para sinais de tempo finito a DFT é uma completa representação de Fourier do sinal. Mais informações descrevendo a DFT e a transformada rápida de Fourier (*fast Fourier transform* – FFT) podem ser obtidas em [21, 42].

3.2.1 Transformada de Fourier de Tempo-Curto

As funções senos e cossenos têm um suporte infinito e são bem adaptadas para analisar sinais estacionários (sinais cujo conteúdo de freqüência não varia no tempo), porém não são apropriados para descrever sinais não-estacionários (transientes), isto é, aqueles nos quais a resposta em freqüência varia no tempo. Nenhuma informação de freqüência está disponível no domínio do tempo do sinal, e nenhuma informação de tempo está disponível no sinal transformado (domínio da freqüência). A transformada de Fourier possui resolução máxima em freqüência mas nenhuma resolução no tempo. Isto significa que podemos determinar todas as freqüências presentes em um sinal, porém não podemos saber quando elas estão presentes.

A transformada de Fourier de tempo-curto (*short-time Fourier transform* – STFT[1]) é uma solução para obter melhor localização no tempo e freqüência na decomposição de um sinal [10, 44]. A STFT é uma versão da transformada de Fourier que utiliza janelas no tempo, e seus respectivos deslocamentos, como bases para a transformada. Em análise de sinais, existem várias escolhas possíveis para a função janela $g(t)$, sendo as principais as que possuem suporte compacto e regularidade razoável. Quando a janela selecionada é Gaussiana, a STFT é conhecida também como transformada de Gabor.

A STFT de um sinal $x(t)$ é definida por:

$$STFT(\tau, f) = \int_{-\infty}^{\infty} [x(t) \cdot g(t-\tau)] \cdot e^{-j2\pi f t}\, dt \qquad (3.5)$$

onde $x(t)$ é o sinal, $g(t\text{-}\tau)$ é a função janela centrado em τ.

Quando baixas freqüências são observadas em um sinal é necessário uma longa observação no tempo. Ao contrário, quando altas freqüências são observadas, somente uma curta observação no tempo é necessária. O princípio da incerteza de Heisenberg, da física quântica, estabelece que não podemos obter a informação exata da freqüência de um sinal e o instante/local exato no tempo/espaço onde esta freqüência ocorreu. O que podemos saber é o intervalo de tempo os quais certas bandas de freqüência existem. Com isso, não é possível obter alta resolução em tempo e freqüência simultaneamente. Em outras palavras, um sinal não pode ser representado como um ponto no espaço tempo-freqüência.

Pelo fato de uma simples janela ser usada para todas as freqüências na STFT, a resolução da análise é a mesma em todas as localizações do plano tempo-freqüência. Uma vantagem da transformada *wavelet* é que a janela varia, com isso teremos funções bases

[1] também conhecida como Transformada de Fourier Janelada (*Windowed Fourier Transform* – WFT)

curtas para alta freqüência e longas para baixa freqüência. A figura 3.1 mostra o diagrama, no plano tempo-freqüência, as bases das funções de Fourier e *Wavelets* [17, 22].

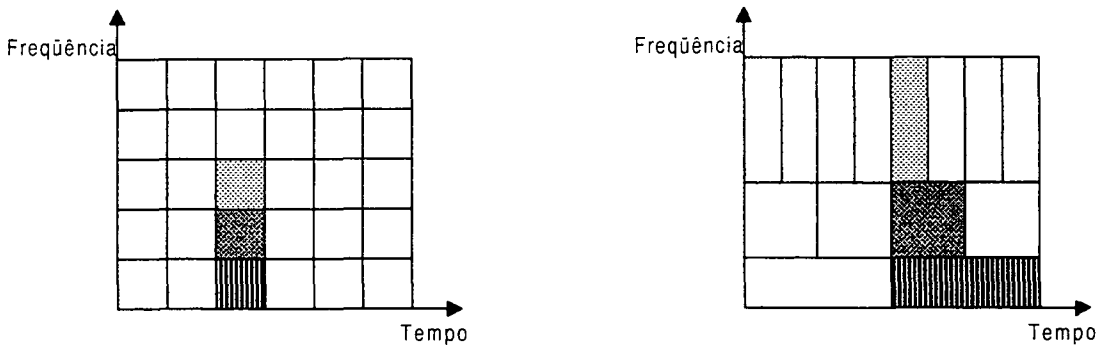


Figura 3.1 – Diagrama do plano tempo-freqüência. A esquerda representação de bases de Fourier e à direita representação de bases *wavelets*.

Podemos observar na figura 3.1 (à direita) que baixas freqüências possuem a altura dos retângulos mais baixas (que correspondem para melhor resolução em freqüência), porém o comprimento é mais longo ("pobre" resolução no tempo). Para altas freqüências ocorre o contrário. Este fato é importante na análise de sinais já que geralmente são os componentes de baixa freqüência que caracterizam o comportamento de um determinado sinal, enquanto que os componentes de alta freqüência nos fornecem os detalhes deste sinal.

3.2.2 Fourier *versus Wavelets*

A transformada de Fourier de tempo-curto permite a análise de um sinal em tempo e freqüência. Já a transformada *wavelet*, que veremos a seguir, permite decompor um sinal em componentes que são bem localizados em tempo (via translação) e escala (via dilatação/contração), introduzindo assim a análise em tempo-escala. No caso de *wavelets*, normalmente não falamos em representação tempo-freqüência mas em representação tempo-escala, porque o termo freqüência é reservado a transformada de Fourier. A partir daqui usaremos o termo tempo-escala. Devido as propriedades de localização em tempo e escala, a transformada *wavelet* pode facilmente detectar informação local em um sinal.

Ao contrário da transformada de Fourier, a transformada *wavelet* não possui um único conjunto de funções base, mas sim vários (infinitos) conjuntos de funções bases (*wavelets*) possíveis.

A transformada discreta *wavelet* (*discrete wavelet transform* – DWT) é uma função ortogonal que pode ser aplicada para um conjunto finito de dados. Ao contrário das funções senos e cossenos da transformada de Fourier, as *wavelets* não precisam ter duração infinita. Este suporte compacto permite a transformada *wavelet* transladar uma função no domínio do tempo em uma representação que não é localizada somente em freqüência (como a transformada de Fourier) mas também no domínio do tempo [8]. O termo translação é usado no mesmo sentido como ele era usado na STFT. Ele está relacionado a localização da janela, quando a janela é deslocada (*shifted*) através do sinal.

Do ponto de vista funcional, a transformada discreta de Fourier é muito parecida com a transformada discreta *wavelet*, naquilo em que a função de transformação é ortogonal, são ambas invertíveis, as matrizes da transformada inversa são as transpostas das originais, o sinal de entrada é assumido ser um conjunto de amostras discretas e ambas as transformadas são convoluções [17, 18].

3.3 A Transformada Wavelet

A transformada *wavelet* é uma ferramenta que permite decompor um sinal em diferentes componentes de freqüências, permitindo assim, estudar cada componente separadamente em sua escala correspondente. O termo "*wavelet*" significa "pequena onda" (*small wave* em inglês ou *ondelette* em francês). O termo "pequena" refere-se à condição de que esta função é de tamanho finito (suportada compactamente).

A transformada *wavelet* contínua (*continuous wavelet transform* – CWT) em $L^2(\mathbf{R})$ pode ser definida como [11, 45]

$$(W_\psi f)(a,b) = \int_{-\infty}^{\infty} f(t).\psi^*_{a,b}(t)dt \qquad (3.6)$$

As funções *wavelets* ($\psi_{a,b}$) são geradas de uma única função $\psi(t)$, denominada *wavelet* mãe (*mother wavelet*) ou *wavelet* básica, através de operações de dilatações e translações definida como

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right) \qquad (3.7)$$

onde $a, b \in \mathbf{R}$, $a \neq 0$, o parâmetro b representa o deslocamento no tempo/espaço, a o fator de escala ($a > 0$ corresponde a dilatação e $a < 0$ para a contração de $\psi(t)$). O fator de multiplicação $\frac{1}{\sqrt{a}}$ é para normalização da energia através das diferentes escalas. Além disso, $\psi, \psi \in L^2(\mathbf{R})$, satisfaz a seguinte condição

$$\int_{-\infty}^{\infty} \psi(t)dt = 0 \qquad (3.8)$$

Para a transformada *wavelet* ser invertível, a função $\psi(t)$ deve satisfazer à condição de admissibilidade

$$C_\psi = \int_{-\infty}^{\infty} \frac{|\hat{\psi}(\xi)|^2}{|\xi|} d\xi < \infty \qquad (3.9)$$

onde $\hat{\psi}$ é a transformada de Fourier,

$$\hat{\psi}(\xi) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{it\xi}\psi(t)dt\,. \qquad (3.10)$$

Se a condição de admissibilidade for satisfeita, a transformada *wavelet* contínua $W(a, b)$ é invertível e a transformada inversa é dada pela relação

$$f(t)=\frac{1}{C_\psi}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}(W_\psi f)(a,b).\psi_{a,b}(t)\frac{dadb}{a^2}. \qquad (3.11)$$

A condição de admissibilidade implica que a transformada de Fourier de $\psi(t)$ se anula na freqüência zero, ou seja

$$\left|\hat{\psi}(\xi)\right|^2\Big|_{\xi=0}=0 \qquad (3.12)$$

e a função *wavelet* $\psi(t)$ deve oscilar. Em outras palavras, $\psi(t)$ deve ser uma onda.

3.3.1 Transformada Discreta *Wavelet*

Na transformada discreta *wavelet* (DWT) os parâmetros de dilatação e translação não variam continuamente, como no caso da transformada *wavelet* contínua, mas sim discretamente. Em certas aplicações, incluindo aquelas em análise de sinal, podemos restringir os valores dos parâmetros a, b (da equação 3.7) a uma grade discreta, fixando um passo de dilatação $a_0 > 1$ e uma passo de translação $b_0 \neq 0$. A família de *wavelets* de interesse, para $j, k \in \mathbf{Z}$, torna-se então [11]

$$\psi_{j,k}(t)=\frac{1}{\sqrt{a_0^{\,j}}}\psi\left(\frac{t-kb_0a_0^{\,j}}{a_0^{\,j}}\right), \text{ ou} \qquad (3.13)$$

$$\psi_{j,k}(t)=a_0^{\,-j/2}\psi(a_0^{\,-j}t-kb_0) \qquad (3.14)$$

Note que isto corresponde para

$$a=a_0^j, \qquad (3.15)$$

$$b=kb_0a_0^j \qquad (3.16)$$

indicando que o parâmetro de translação b depende da taxa de dilatação escolhida. Para j grande e positivo, a função $\psi_{j,0}$ é bastante dilatada, e os passos de translação grandes ($b_0a_0^j$)

são adaptados a esta grande largura. Para j grande e negativo ocorre o contrário; a função $\psi_{j,0}$ é bastante contraída e os passos de translação pequenos $b_0 a_0^j$ são necessários para ainda cobrir toda a extensão.

Se assumirmos dilatações binárias e translações unitárias, isto é, $a_0 = 2$ e $b_0 = 1$, a função *wavelet* torna-se

$$\psi_{j,k}(t) = 2^{-j/2}\psi(2^{-j}t - k) \qquad (3.17)$$

e constitui uma base ortonormal para $L^2(\mathbf{R})$. Dessa forma, teremos uma amostragem diádica que é mais adequada para cálculos computacionais.

3.4 Análise em Multiresolução

O conceito de análise em multiresolução, desenvolvido por Mallat [32], permite analisar um sinal em diferentes freqüências com diferentes resoluções. Com a análise em multiresolução é possível obter uma boa resolução no tempo e pobre resolução em freqüência, que se torna útil pelo fato de que os sinais encontrados em aplicações práticas geralmente apresentam componentes de alta freqüência por curtas durações de tempo e componentes de baixa freqüência por longa duração de tempo.

Uma análise em multiresolução de $L^2(\mathbf{R})$ é definida por uma seqüência de subespaços fechados $V_j \subset L^2(\mathbf{R}), j \in \mathbf{Z}$, satisfazendo as seguintes propriedades [9, 29, 32, 45, 63]:

a) Propriedade de aninhamento

$$\cdots \subset V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \cdots \qquad (3.18)$$

b) Densidade da união em $L^2(\mathbf{R})$

$$\bigcup_{j \in Z} V_j \text{ é denso em } L^2(\mathbf{R}) \text{ e} \qquad (3.19)$$

$$\bigcap_{j \in \mathbf{Z}} V_j = \{0\} \qquad (3.20)$$

c) Propriedade de escala

$$f(t) \in V_j \Leftrightarrow f(2t) \in V_{j+1} \qquad \forall j \in \mathbf{Z} \qquad (3.21)$$

d) Propriedade de invariância do deslocamento

$$f(t) \in V_0 \Rightarrow f(t - n) \in V_0 \qquad \forall_n \in \mathbf{Z} \qquad (3.22)$$

e) Existência de uma função de escala

$$\exists \varphi \in V_o \text{ tal que } \{\varphi(t-k) \mid k \in \mathbf{Z}\} \text{ é uma base de } Riesz \text{ de } V_0. \qquad (3.23)$$

Se $\{\varphi(t - k) \mid k \in \mathbf{Z}\}$ é uma base ortonormal para V_0, temos uma análise em multiresolução e as bases *wavelet* construídas de $\varphi(t)$ são chamadas *wavelets* ortonormais. A função φ é chamada de função de escala e é usada para construir bases *wavelets*. Multiresolução requer uma base para cada espaço V_j. Para os outros subespaços V_j (para $j \neq 0$) definimos

$$\varphi_{j,k}(t) = 2^{j/2}\varphi(2^j t - k) \qquad (3.24)$$

onde o índice j denota a escala e k indica o deslocamento inteiro.

Desde que $\varphi \in V_0 \subset V_1$, então existe um conjunto finito de coeficientes h_k tal que a função de escala satisfaz

$$\varphi(t) = \sum_{k \in Z} h_k \varphi_{1,k}(t) = \sqrt{2} \sum_{k \in Z} h_k \varphi(2t - k). \qquad (3.25)$$

Esta equação é conhecida por vários nomes diferentes: equação de refinamento (*refinement equation*), equação de dilatação (*dilation equation*) ou equação de diferença de escala-dois (*two-scale difference equation*).

A função $\varphi(t)$ é usualmente normalizada, então temos

$$\int_{-\infty}^{\infty} \varphi(t)\, dt = 1. \qquad (3.26)$$

Uma base de *Riesz* de um espaço de Hilbert H é um subconjunto $\{e_1, e_2, ..., e_n, ...\}$ de H tal que $[e_1, e_2, ..., e_n, ...]$ é denso em H e existem constantes $0 < A < B < \infty$ tais que para toda seqüência de escalares $\alpha_0, \alpha_1, \alpha_2, ...$ tem se: [9, 19]

$$A\left(\sum_{k=0}^{+\infty}|\alpha_k|^2\right) \le \left\|\sum_{k=0}^{+\infty}\alpha_k e_k\right\|^2 \le B\left(\sum_{k=0}^{+\infty}|\alpha_k|^2\right) \qquad (3.27)$$

3.5 Funções Wavelets

Como temos que o subespaço $V_j \subset V_{j+1}$, podemos definir W_j, $\forall j \in \mathbf{Z}$, como o complemento ortogonal de V_j em V_{j+1}, isto é, um espaço que satisfaz [9, 29, 45]

$$V_{j+1} = V_j \oplus W_j \qquad W_j \perp V_j \qquad (3.28)$$

onde o símbolo $\oplus$ é a soma direta, e $\forall u \in V_{j+1}$, $u = v + w$, de modo que $v \in V_j$ e $w \in W_j$, com $\langle v, w \rangle = 0$, ou seja, cada elemento de V_{j+1} pode ser escrito, de maneira única, como uma soma de um elemento de W_j com um elemento de V_j.

O espaço W_j contém as informações detalhe, necessárias para ir de V_j em V_{j+1}. Estas informações detalhe são extraídas do sinal original usando a função *wavelet* $\psi(t)$. Consequentemente

$$V_{j+1} = \bigoplus_{-\infty}^{j} W_j, \quad \text{e} \quad L^2(\mathbf{R}) = \bigoplus_{j\in\mathbf{Z}} W_j \qquad (3.29)$$

onde todos os subespaços W_j são mutualmente ortogonais, isto é

$$W_j \perp W_k, \quad j \neq k. \qquad (3.30)$$

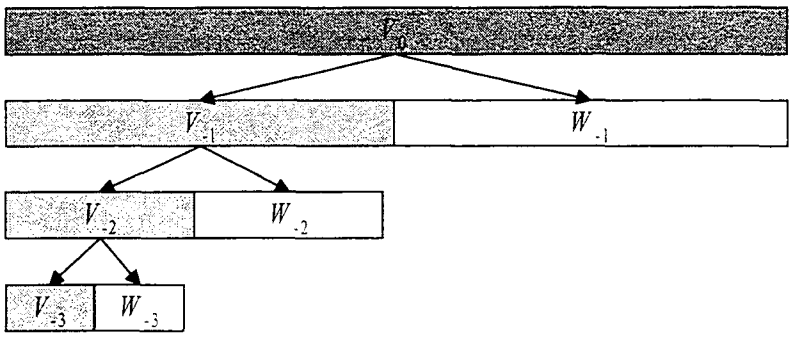


Figura 3.2 – Árvore de decomposição da transformada *wavelet*.

A função ψ é uma *wavelet* se o conjunto de funções $\{\psi(t-k) \mid k \in \mathbf{Z}\}$ é uma base de *Riesz* de W_0 [63]. A coleção de funções *wavelet* $\{\psi_{j,k} \mid j, k \in \mathbf{Z}\}$ é então uma base de *Riesz* de $L^2(\mathbf{R})$.

Uma função $\psi(t) \in L^2(\mathbf{R})$ tal que $\{\psi(t-k) \mid k \in \mathbf{Z}\}$ é dita ser ortonormal se as funções $\psi_{j,k}$ formarem uma base ortonormal para W_0. Para os outros subespaços W_j (para $j \neq 0$) e $k \in \mathbf{Z}$, definimos

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k). \qquad (3.31)$$

Esta função *wavelet* diádica é um tipo particular de transformada *wavelet* (outras dilatações são possíveis, porém dilatações diádicas são mais práticas computacionalmente). Para a transformada unidimensional, *wavelets* $\psi_{j,k}$, são geradas por escalonamento binário (contraindo por um fator de 2) e translações diádicas de uma *wavelet* $\psi(t)$.

Pelo fato de que V_0 e W_0 são subespaços de V_1, $V_0 \subset V_1$ e $W_0 \subset V_1$, podemos expressar a *wavelet* $\psi(x)$ em termos da função de escala $\varphi(t)$

$$\psi(t) = \sum_{k \in Z} g_k \varphi_{1,k}(t) = \sqrt{2}\sum_{k \in Z} g_k \varphi(2t-k), \qquad (3.32)$$

para um conjunto finito de coeficientes g_k, de modo que

$$g_k = (-1)^k h_{1-k}, \qquad (3.33)$$

satisfazendo

$$\sum_k g_k = 0. \qquad (3.34)$$

Vimos até momento os aspectos teóricos da transformada *wavelet*, todavia tudo isso seria de pouca importância prática se não fosse pelo fato de podermos computar eficientemente os coeficientes *wavelets* e reconstruir funções a partir desses coeficientes. Estes algoritmos, conhecidos como transformada *wavelet* rápida (*fast wavelet transform* – FWT), são análogos a transformada rápida de Fourier e seguem da equação de refinamento mencionada acima.

Veremos a seguir a função *wavelet* de Haar, uma função ortonormal que apresenta suporte compacto e é a forma mais simples de *wavelets*. A função de Haar é gerada diretamente da função mãe e constitui uma base ortonormal de $L^2(\mathbf{R})$.

3.6 A Função Wavelet de Haar

A função *wavelet* de Haar, o qual será discutido em mais detalhes nesta seção, é o exemplo mais simples de *wavelets*. A transformada de Haar foi introduzida em 1910, por Alfred Haar [24], e é o mais antigo dos métodos de transformada *wavelet*. A transformada de Haar usa pulsos quadrados para aproximar a função original. Transformadas *wavelet* utilizando funções de Haar, como funções bases, são as mais simples para implementar e são computacionalmente as menos exigentes.

Na construção de bases *wavelets* normalizada, o filtro passa baixa de Haar h_k é definido por $h_0 = h_1 = \frac{1}{\sqrt{2}}$ e todos os demais coeficientes iguais a zero. Substituindo o filtro passa baixa de Haar na equação 3.25, teremos

$$\varphi(t) = \varphi(2t) + \varphi(2t - 1) \qquad (3.35)$$

A solução para esta recorrência é a função de escala de Haar

$$\varphi(t) = \begin{cases} 1 & \text{para } 0 \le t < 1 \\ 0 & \text{caso contrário} \end{cases} \qquad (3.36)$$

As funções de escala de Haar são mostradas na figura abaixo.

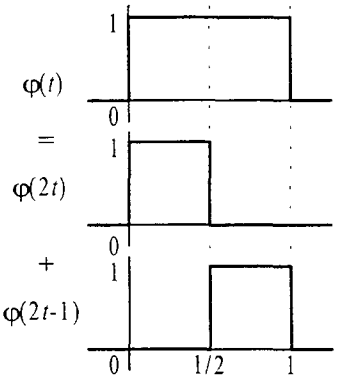


Figura 3.3 – Funções de escala de Haar $\varphi(t)$, $\varphi(2t)$ e $\varphi(2t - 1)$.

O filtro passa alta de Haar g_k é definido por $g_0 = \frac{1}{\sqrt{2}}$, $g_1 = -\frac{1}{\sqrt{2}}$ e zero para todos os demais coeficientes. Substituindo o filtro passa alta de Haar na equação 3.32, teremos

$$\psi(t) = \varphi(2t) - \varphi(2t - 1) \qquad (3.37)$$

A solução para esta recorrência é a função *wavelet* de Haar

$$\psi(t) = \begin{cases} 1 & \text{para } 0 \le t < \frac{1}{2} \\ -1 & \text{para } \frac{1}{2} \le t < 1 \\ 0 & \text{caso contrário} \end{cases}$$

As função *wavelet* $\psi(t)$ é mostrada na figura abaixo.

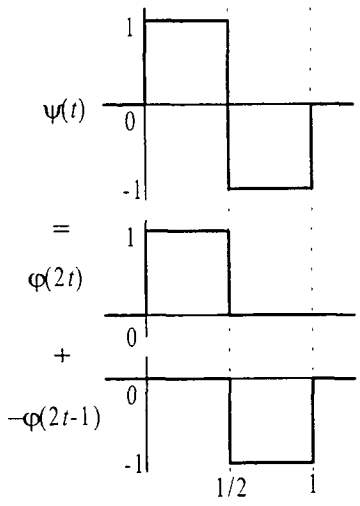


Figura 3.4 – Funções *wavelet* de Haar $\psi(t)$, $\varphi(2t)$ e $-\varphi(2t-1)$.

Aplicando $\varphi(t)$ para $f(t)$ teremos

$$<\varphi, f> = \int_{-\infty}^{\infty} \varphi(t) f(t)\, dt = \int_{0}^{1} f(t)\, dt\,, \qquad (3.38)$$

o valor da média de f sobre o intervalo [0, 1). Aplicando $\psi(t)$ para $f(t)$ teremos

$$<\psi, f> = \int_{-\infty}^{\infty} \psi(t) f(t)\, dt = \int_{0}^{1/2} f(t)\, dt - \int_{1/2}^{1} f(t)\, dt\,. \qquad (3.39)$$

O filtro φ é um operador de média que serve para suavizar o sinal e o filtro ψ é um operador diferença que é utilizado para reconstruir o sinal.

Segue abaixo as rotinas, codificadas na linguagem C, para calcular a transformada de Haar e sua inversa, de um sinal unidimensinal armazenado no vetor *M*. A decomposição do sinal é efetuado $\log_2(size)$ vezes, que corresponde a transformar o sinal até obter a menor resolução possível. O número de amostras do sinal é definido em *size*, os quais deve ser potência de 2.

```
void Haar(int size)
{
  int i;
  while (size>1)
  {
    for (i=0; i<size/2; i++)
```

```
    {
      S[i]=(M[2*i]+M[2*i+1])/2;          /* filtro passa baixa (média) */
      S[size/2+i]=(M[2*i]-M[2*i+1])/2;   /* filtro passa alta (diferença) */
    }
    for (i=0; i<size; i++)
      M[i]=S[i];
    size=size/2;
  }
}

void InvHaar(int size)
{
  int i,j=1;
  while (j<size)
  {
    for (i=0; i<j; i++)
    {
      M[2*i]=S[i]+(S[j+i]);          /* recupera as amostras pares */
      M[2*i+1]=S[i]-(S[j+i]);        /* recupera as amostras impares */
    }
    j=j*2;
    for (i=0; i<j; i++)
      S[i]=M[i];
  }
}
```

Computacionalmente, a transformada de Haar tem uma complexidade linear, requerendo apenas O(n) operações aritméticas. De fato, se aplicarmos o banco de filtros de síntese para um sinal com n amostras será necessário kn operações para alguma constante k, então o número total de operações para computar a transformada *wavelet* será

$$kn + k\frac{n}{2} + k\frac{n}{4} + \cdots + k\frac{n}{\log_2 n} \leq kn\left(1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{\log_2 n}\right) \leq 2kn = O(n). \qquad (3.40)$$

A cada etapa do processo de decomposição hierárquica o custo computacional é reduzido pela metade, com exceção da primeira etapa, decorrente da subamostragem por 2. O custo inicial é de kn operações, que será reduzido para $k\frac{n}{2}$ que corresponde a metade da etapa anterior, porque utilizamos apenas a metade dos coeficientes (coeficientes passa baixa), e assim por diante, limitando a um custo computacional de no máximo $2kn$ operações.

3.7 Codificação Sub-banda e Banco de Filtros

Uma representação em tempo-escala de um sinal digital é obtido usando técnicas de filtragens digitais [44]. Filtrar um sinal corresponde a operação matemática de convolução do sinal com a resposta ao impulso do filtro. A CWT era computada mudando-se a escala da janela em análise, deslocando a janela no tempo, multiplicando pelo sinal, e integrando sobre todo o tempo. No caso discreto, filtros de diferentes freqüência de corte são usados para analisar o sinal em diferentes escalas. O sinal é passado através de uma série de filtros passa alta para analisar as altas freqüências, e através de uma série de filtros passa baixa para analisar as baixas freqüências.

3.7.1 Redução na Taxa de Amostragem por um Fator Inteiro

Subamostragem (decimador) de um sinal corresponde a reduzir a taxa de amostragem (*subsampling*), ou remover algumas amostras do sinal. Por exemplo, subamostrar um sinal de entrada por um fator de 2 refere-se a extrair uma amostra a cada duas, de modo que o sinal de saída terá a metade do número de amostras. Subamostragem de um sinal por um fator M reduz o número de amostras do sinal M vezes, ou seja $x_d[n] = x[nM] = x_c(nMT)$.

A representação do símbolo do subamostrador é visto abaixo [42].

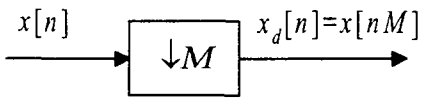


Figura 3.5 – Representação de um subamostrador de fator M.

3.7.2 Aumento na Taxa de Amostragem por um Fator Inteiro

Superamostragem (interpolação) de um sinal corresponde a aumentar a taxa de amostragem (*upsampling*) de um sinal adicionando novas amostras para o sinal. Por exemplo, subamostrar um sinal por 2 refere-se a adicionar uma nova amostra, usualmente zero ou um valor interpolado, entre cada duas amostras do sinal. Superamostragem de um sinal por um fator L aumenta o número de amostras do sinal L vezes.

A representação do símbolo do superamostrador é visto abaixo [42].

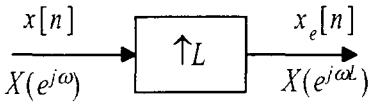


Figura 3.6 – Representação de um superamostrador de fator L.

3.7.3 Esquema de Codificação Sub-banda

A DWT emprega dois conjuntos de funções chamadas funções de escala e funções *wavelet*, os quais são associadas com filtros passa baixa e filtros passa alta respectivamente. A DWT é computada analisando o sinal em diferentes bandas de freqüências com diferentes resoluções através da decomposição do sinal em componentes aproximação (ou *smooth*) e componentes detalhe. São os componentes detalhe que armazenam as informações necessárias para permitir a reconstrução da imagem a partir dos componentes aproximação.

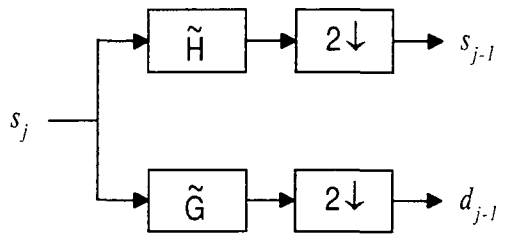


Figura 3.7 – Codificação sub-banda de análise.

A decomposição do sinal em diferentes bandas de freqüências é obtida simplesmente por sucessivas filtragens passa baixa e passa alta do sinal no domínio do tempo. Filtros passa baixa e passa alta juntos constituem um banco de filtros. O sinal original s_j (figura 3.7) é primeiro filtrado por um filtro passa baixa $\tilde{H}$ e um filtro passa alta $\tilde{G}$. Depois da filtragem do sinal, a metade das amostras podem ser eliminadas por um subamostrador de fator 2 ($2\downarrow$). O resultado será um sinal passa baixa (s_{j-1}) e um sinal passa alta (d_{j-1}), cada um deles contendo a metade das amostras do sinal de entrada s_j.

Em processamento digital de sinais, os filtros $\tilde{H}$ e $\tilde{G}$ são chamados de filtros de quadratura espelhada (*quadrature mirror filters* – QMF). Esses filtros foram estudados antes da teoria *wavelet*.

Esta decomposição reduz pela metade a resolução no tempo dado que somente a metade do número de amostras agora caracterizam o sinal de entrada. O procedimento acima, que é também conhecido como codificação por sub-banda, pode ser repetida para obter uma decomposição adicional [44].

Podemos construir uma representação hierárquica de um sinal filtrando recursivamente a saída passa baixa do banco de filtros. Este processo é ilustrado graficamente na figura 3.8 para quatro níveis de resolução (os filtros aplicados quatro vezes), onde s_j é o sinal original que será decomposto (ou transformado) e $\tilde{H}$ e $\tilde{G}$ são filtros passa baixa e passa alta, respectivamente. Este esquema é conhecido também como decomposição piramidal.

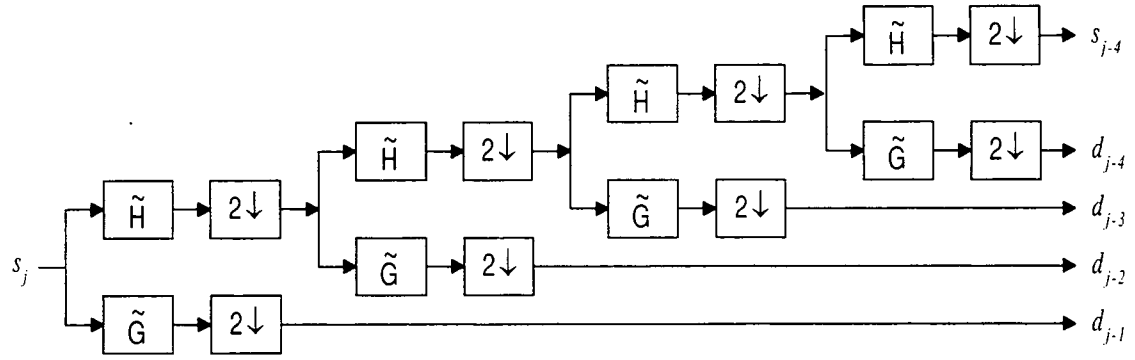


Figura 3.8 – Banco de filtros de análise hierárquico.

A saída s_{j-4} do banco de filtros é uma versão do sinal de entrada numa resolução dezesseis vezes menor. O filtro passa alta produz os coeficientes *wavelets* para o nível, e o filtro passa baixa produz a função escala para o próximo nível da decomposição hierárquica.

Os coeficientes da saída s_{j-4} corresponde ao subespaço V_{-4} e os coeficientes d_{j-4}, d_{j-3}, d_{j-2} e d_{j-1}, correspondem aos subespaços W_{-4}, W_{-3}, W_{-2} e W_{-1} respectivamente. O conjunto de aproximações sucessivas (V_j) juntamente com o conjunto de detalhes sucessivos (W_j) forma o que chamamos de decomposição em multiresolução do sinal original.

Nos sistemas ditos de dois canais, o sinal de entrada é transformado em duas bandas, sendo uma de baixa freqüência e outra de alta freqüência. Quando as bandas de baixa freqüência forem entradas para um outro sistema de banco de filtros, idêntico ao primeiro, cria-se uma estrutura do tipo árvore, como na figura 3.8, que divide o espectro do sinal original em oitavas. A figura 3.9 ilustra a decomposição do sinal em oitavas, que é o ponto inicial do esquema em multiresolução.

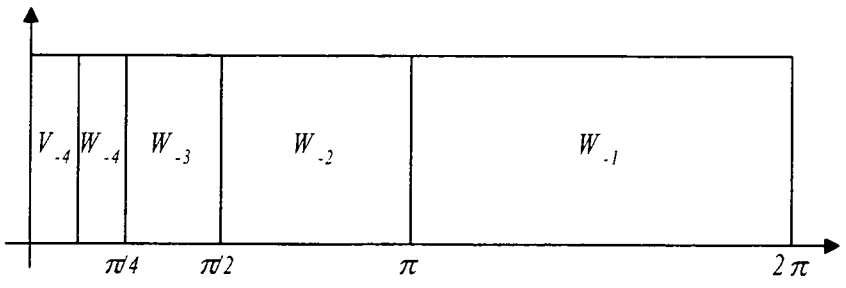


Figura 3.9 – Decomposição do espectro em oitava.

O processo de síntese, ou reconstrução, consiste em interpolar o sinal por um fator de 2 (↑2), ou seja, colocar zeros entre cada amostra. Em seguida, o sinal é filtrado utilizando um filtro passa baixa H e um filtro passa alta G inversos aos filtros $\tilde{H}$ e $\tilde{G}$, respectivamente. Uma representação do processo de síntese, utilizando um banco de filtros de dois canais é ilustrado abaixo:

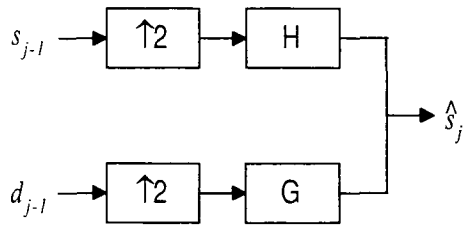


Figura 3.10 – Codificação sub-banda de síntese.

A figura abaixo ilustra o processo de reconstrução hierárquica de um sinal que foi transformado em quatro níveis de resolução.

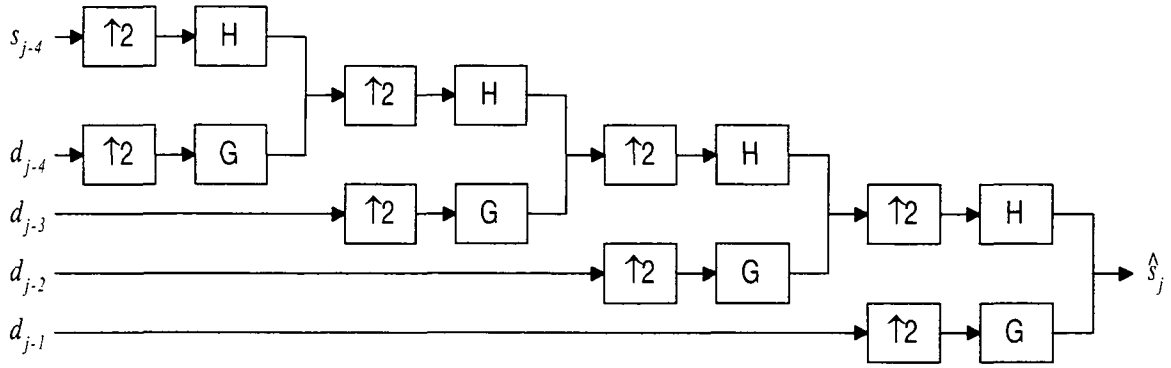


Figura 3.11 – Banco de filtros de síntese hierárquico.

Um banco de filtro digital é uma coleção de filtros digitais, com uma entrada ou uma saída comum. O esquema completo para um banco de filtros com dois canais é descrito abaixo na figura 3.12. Ele envolve dois filtros de análise $\tilde{H}$ (passa baixa) e $\tilde{G}$ (passa alta) e dois filtros de síntese H (passa baixa) e G (passa alta).

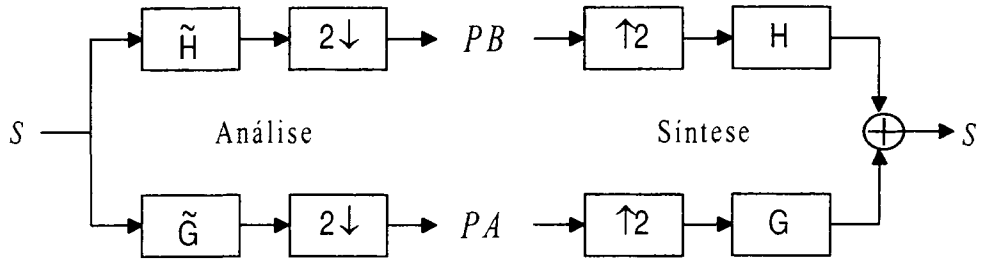


Figura 3.12 – Banco de filtros com dois canais.

Uma perfeita reconstrução do sinal de entrada S pode ser obtida caso seja realizado um projeto apropriado dos filtros de análise e filtros de síntese do banco de filtros. Um banco de filtros é dito ser um banco de filtros de reconstrução perfeita (*perfect reconstruction filter bank* – PRFB) se o sinal de saída for igual ao sinal de entrada, como na figura 3.12, e teremos nesse caso, os filtros de análise seguido pelos de síntese igual a identidade

$$H\tilde{H} + G\tilde{G} = I, \tag{3.41}$$

e os filtros de síntese seguido pelos de análise é a identidade também, ou seja

$$\tilde{H}H = I, \quad \tilde{H}G = 0, \quad \tilde{G}H = 0, \quad \tilde{G}G = I. \tag{3.42}$$

A teoria de banco de filtros estabelece que para eliminar *aliasing* a relação [66]

$$g_1(n) = (-1)^{n+1}h_0(n) \quad \text{e} \quad h_1(n) = (-1)^n g_0(n) \tag{3.43}$$

devem ser satisfeitas, onde $g_0(n)$ e $g_1(n)$ são os filtros de síntese e $h_0(n)$ e $h_1(n)$ os filtros de análise. Neste caso, temos os filtros de síntese definidos em termos dos filtros de análise.

3.8 Transformada Wavelet Aplicado à Imagens

A figura 3.13 mostra os processos de compressão e descompressão aplicados em imagens digitais. A perda de informações geralmente ocorrem na aplicação da transformada *wavelet* (decorrentes do arredondamentos de valores de ponto flutuante) e na etapa de quantização/limiar (*threshold*) dos coeficientes transformados. Os codificadores de entropia permitem a compressão sem que haja a perda de informações.

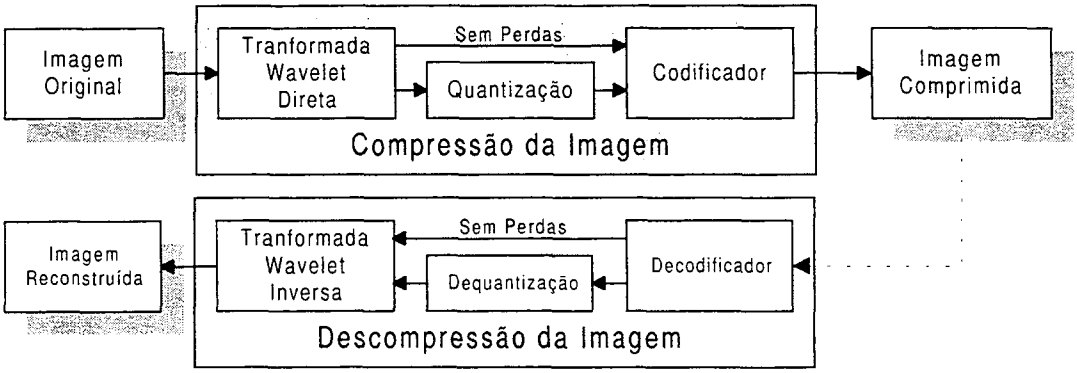


Figura 3.13 – Diagrama em blocos do processo de compressão/descompressão de imagens.

A transformada *wavelet* direta mapeia os dados da imagem original para um outro domínio, sem fornecer nenhuma compressão dos dados em relação a imagem original, porém a transformada inversa, em muitos casos, permite uma reconstrução exata das informações anteriores. Neste novo domínio os dados são caracterizados por uma grande quantidade de valores iguais ou próximos de zero, que torna eficiente o uso de codificadores de entropia. A compressão é realizada pela quantização/limiar e pela codificação dos coeficientes *wavelets*. A reconstrução da imagem é efetuada invertendo as operações do processo de compressão.

Considerando novamente a transformada *wavelet* de Haar não normalizada, demonstraremos como multiplicações de matrizes podem ser utilizadas para efetuar as médias e diferenças. Seja A_1 uma matriz 8 × 8 formada pela base de Haar e assumindo M um vetor com 8 elementos correspondente aos dados de entrada e contido em V_0.

Multiplicando a matriz M pela matriz A_1 obteremos a matriz M', que corresponde ao primeiro nível de decomposição da matriz de entrada. Isto significa que os componentes de entrada é separado em duas seqüências, uma representado as médias e outra as diferenças. A matriz A_1 é mostrada a seguir [37].

$$A_1 = \frac{1}{2} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix}$$

A operação acima corresponde à decomposição do subespaço V_0 em V_{-1} e W_{-1}. Em seguida o subespaço V_{-1}, que corresponde aos coeficientes média, deverá ser transformado. Para isso, deveremos multiplicar a matriz M' pela matriz A_2 resultando na matriz M''. A multiplicação das matrizes alterará apenas os valores da primeira metade do vetor. Esta operação corresponde à decomposição do subespaço V_{-1} em V_{-2} e W_{-2}. A matriz A_2 e mostrada a seguir.

$$A_2 = \frac{1}{2} \cdot \left[\begin{array}{cccc:cccc} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{array}\right]$$

Finalmente, a matriz M'' deverá ser multiplicado por A_3. Como a decomposição é binária, podemos ter, no máximo $\log_2 8$, isto é, 3 níveis de decomposição diádica. Chamaremos de N o resultado do produto da matriz de entrada pelas matrizes A_1, A_2 e A_3, que consiste em decompor o subespaço V_0 em V_{-3}, W_{-3}, W_{-2} e W_{-1}. A matriz A_3 é mostrada a seguir.

$$A_3 = \frac{1}{2} \cdot \left[\begin{array}{cccc:cccc} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{array}\right]$$

A seqüência final formada pelo vetor N corresponde ao sinal original transformado. O primeiro elemento dessa seqüência é chamado de coeficiente de escala (ou aproximação) e os demais elementos dessa seqüência são os coeficientes detalhe, ou coeficientes *wavelets*.

Este processo aplicado à seqüência original para a obtenção de versões de mais baixa resolução é chamado de análise ou decomposição e o processo para a obtenção dos coeficientes da transformada é conhecido como transformada *wavelet*.

Chamaremos de W a matriz formada pela multiplicação das matrizes A_1, A_2 e A_3, respectivamente. Uma representação da matriz W é apresentada a seguir.

$$W = \frac{1}{8} \cdot \begin{bmatrix} 1 & 1 & 2 & 0 & 4 & 0 & 0 & 0 \\ 1 & 1 & 2 & 0 & -4 & 0 & 0 & 0 \\ 1 & 1 & -2 & 0 & 0 & 4 & 0 & 0 \\ 1 & 1 & -2 & 0 & 0 & -4 & 0 & 0 \\ 1 & -1 & 0 & 2 & 0 & 0 & 4 & 0 \\ 1 & -1 & 0 & 2 & 0 & 0 & -4 & 0 \\ 1 & -1 & 0 & -2 & 0 & 0 & 0 & 4 \\ 1 & -1 & 0 & -2 & 0 & 0 & 0 & -4 \end{bmatrix}$$

Como as colunas das matrizes A_i são ortogonais, cada uma destas matrizes são inversíveis. Denotaremos por W^{-1} a transformada *wavelet* inversa de Haar, onde $W^{-1} = A_3^{-1}.A_2^{-1}.A_1^{-1}$. A matriz W^{-1} é mostrada abaixo.

$$W^{-1} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

Seja M a matriz de uma imagem $2^r \times 2^r$, então a equação $Q = MW$ e $M = QW^{-1}$ expressam a relação entre M e as linhas da imagem transformada Q, onde $W = A_1 . A_2 . \ldots . A_r = \prod_{i=1}^{r} A_i$. Para efetuar a transformação nas colunas da imagem, devemos repetir os passos acima com a matriz transposta. Teremos então a seguinte equação, os quais expressa a relação entre a imagem original M e a imagem transformada nas linhas e colunas N:

$$N = ((M.W)^T)^T = W^T.M.W \qquad (3.44)$$

e a equação de composição, ou transformada inversa, dado por

$$M = ((N^T).W^{-1})^T.W^{-1} = (W^{-1})^T.N.W^{-1} \qquad (3.45)$$

A aplicação da transformada bidimensional de Haar pode ser vista como a aplicação da transformada de Haar em uma das dimensões e em seguida na outra dimensão da imagem. Este fato é resultante de uma das propriedades da transformada de Haar, conhecida por separabilidade. Esta propriedade de separabilidade é uma característica importante, o qual faz da transformada *wavelet* uma poderosa ferramenta em processamento de sinais em várias dimensões. Para um sinal com uma dimensão n maior do que 1, a transformada *wavelet* é realizada através da transformação de cada dimensão do sinal independentemente.

A transformada *wavelet* de Haar normalizada, mencionada anterior, pode ser implementada substituindo as constantes iguais a $\frac{1}{2}$ e todos os coeficientes iguais a 2, das matrizes A_i, por $\frac{1}{\sqrt{2}}$ e $\sqrt{2}$, respectivamente. As colunas de cada matriz A_i formam então uma base ortonormal. Consequentemente, o mesmo é verdade para a matriz W.

3.8.1 Transformada *Wavelet* Bidimensional

Existem duas maneiras pelo qual podemos usar *wavelets* para decompor uma imagem bidimensional [57]: decomposição padrão e decomposição não-padrão.

Para obter a decomposição padrão de uma imagem, primeiro aplicamos a transformada *wavelet* unidimensional em cada uma das linhas da imagem. Esta operação fornece um coeficiente de média e os coeficientes detalhe para cada linha. Em seguida, tratamos estas linhas transformadas como se elas fossem uma nova imagem e aplicamos mais uma vez a transformada unidimensional para cada coluna da imagem. O resultado destas operações serão todos coeficientes detalhe, exceto o primeiro *pixel*, que corresponde ao único coeficiente aproximação (média). Neste caso, estamos levando em consideração a aplicação da transformada até obter o menor nível de resolução possível.

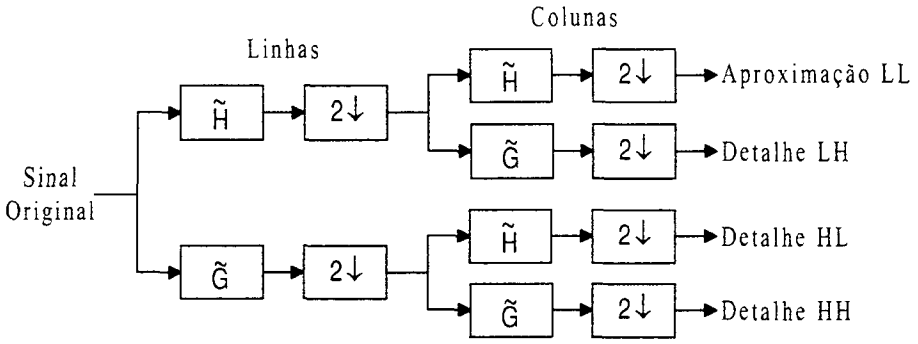


Figura 3.14 – Transformada discreta *wavelet* bidimensional.

Dado uma imagem $x(m, n)$, ela é inicialmente filtrada na direção m (linhas da imagem), resultando numa imagem passa baixa L e uma imagem passa alta H. Após a subamostragem, teremos ambas as imagens reduzidas pela metade em relação a imagem original. Em seguida realiza-se a filtragem na direção n (colunas da imagem) resultando em quatro subimagens:

- LL (passa baixa-passa baixa) correspondendo a banda passa baixa em ambas as direções.
- LH (passa baixa-passa alta) correspondendo a banda passa baixa na direção vertical e passa alta na direção horizontal.
- HL (passa alta-passa baixa) correspondendo a banda passa alta na direção vertical e passa baixa na direção horizontal, e
- HH (passa alta-passa alta) correspondendo a banda passa alta em ambas as direções.

A aplicação da transformada *wavelet* na imagem resulta em uma imagem com o mesmo tamanho mas composto de três imagens detalhes (HL, LH e HH) e uma imagem aproximação (LL), sendo que todas possuem a metade da resolução da imagem inicial.

Este processo pode ser repetido novamente na subimagem LL, resultando em mais quatro subimagens, e assim por diante até que tenhamos apenas um único coeficiente aproximação. O menor nível de resolução possível conterá um único coeficiente os quais é a média de todas as amostras do sinal original, isto é, ele é o coeficiente DC ou freqüência zero do sinal. O algoritmo da transformada *wavelet* inversa é construído de maneira semelhante aplicando o processo inverso.

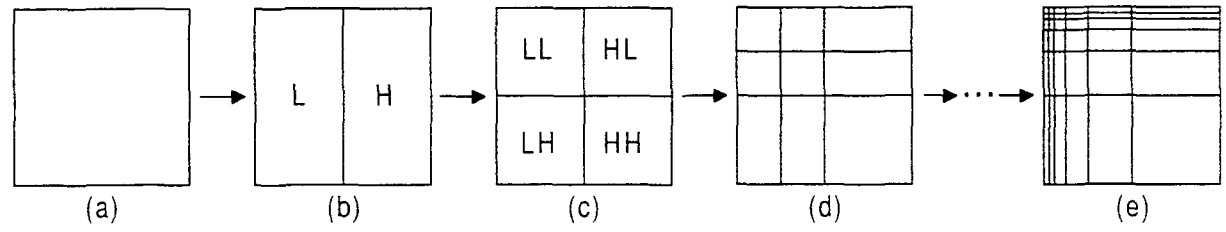


Figura 3.15 – Estágios de decomposição *wavelet* bidimensional padrão com 5 níveis de resolução.

A seguir mostraremos a aplicação da transformada *wavelet* padrão, usando a imagem Lenna. Inicialmente a transformada é aplicado apenas nas linhas da imagem, e em seguida apenas nas colunas.

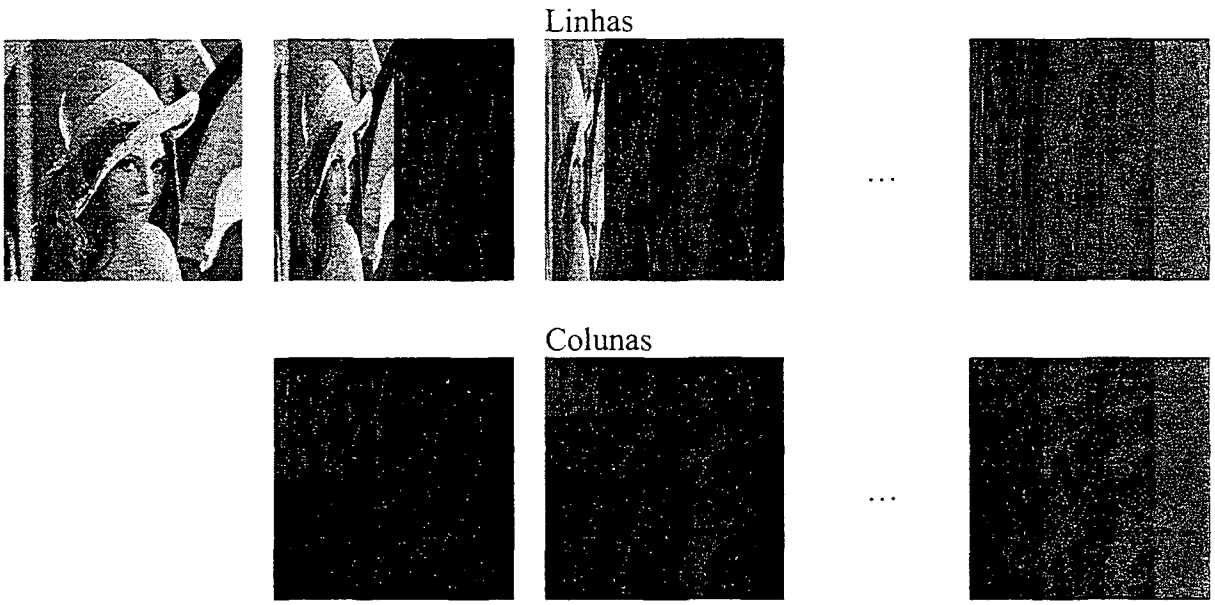


Figura 3.16 – Decomposição padrão da imagem Lenna.

O segundo tipo de transformada *wavelet* bidimensional, chamada decomposição não padrão, alterna entre operações nas linhas e nas colunas. Primeiro, calculamos um passo de médias e diferenças no valor dos *pixels* de cada linha da imagem. Em seguida, calculamos as médias e as diferenças em cada coluna. Após esta operação obteremos a imagem da figura 3.17 (c), composta por quatro imagens menores. A imagem do canto superior esquerdo contém os coeficientes de baixa resolução, correspondente à média dos *pixels* da imagem

original, enquanto que as três demais imagens contêm os coeficientes de alta resolução, os coeficientes *wavelets*, que irão permitir a reconstrução da imagem. Para completar a transformação, repetimos este processo recursivamente, somente nos quadrantes contendo as médias, em ambas direções.

Se uma imagem for decomposta em N níveis de resolução ($N > 0$ e $N \in \mathbf{Z}$), isso implicará na obtençao de $3N + 1$ subimagens.

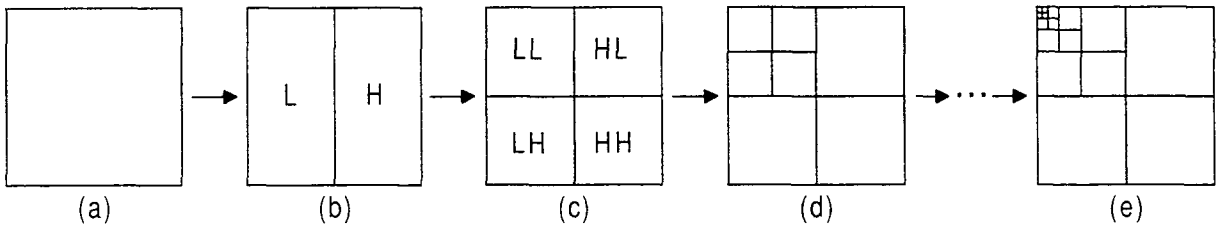


Figura 3.17 – Estágios de decomposição *wavelet* bidimensional não padrão.

A figura 3.17 ilustra os cinco primeiros níveis de resolução de uma imagem bidimensional após a aplicação da transformada *wavelet* não padrão. Em (a) temos a imagem original, (b) a imagem após a aplicação da transformada em suas linhas, (c) aplicação da transformada nas linhas e nas colunas, um nível de resolução, (d) a imagem com dois níveis de resolução e finalmente em (e) a imagem após cinco níveis de resolução.

A seguir mostraremos a aplicação da decomposição *wavelet* não padrão, usando a imagem Lenna.

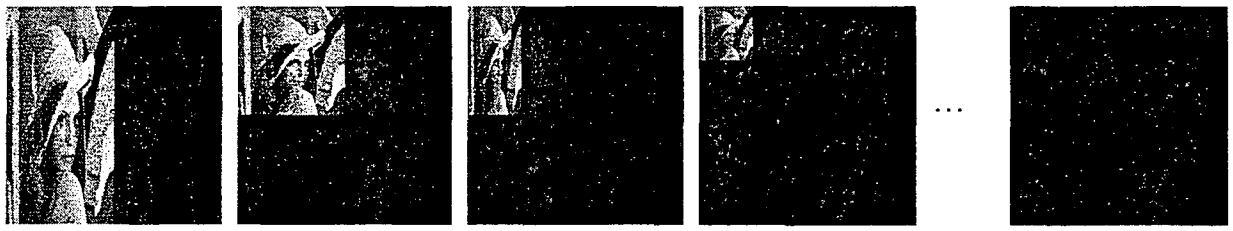


Figura 3.18 – Decomposição não padrão da imagem Lenna.

A decomposição *wavelet* padrão de uma imagem é mais atraente pelo fato de requerer somente operações em uma dimensão, ou seja, primeiro aplica-se a transformada apenas nas linhas e em seguida apenas nas colunas da imagem. Por outro lado, é um pouco

mais eficiente a computação da decomposição não-padrão. Para uma imagem $m \times m$, a decomposição padrão requer o cálculo de $4(m^2 - m)$ operações, enquanto que a decomposição não padrão requer somente $\frac{8}{3}(m^2 - 1)$ operações [57].

Podemos verificar que a transformada *wavelet* permite armazenar uma imagem em diversas resoluções, ilustrado também na figura 3.19. Dessa maneira, podemos transmitir inicialmente os coeficientes da imagem com menor resolução, permitindo assim a visualização de uma aproximação da imagem. Com isso, é possível efetuar a reconstrução gradual da imagem pelo receptor. Em seguida, somente a informação necessária para derivar uma versão mais detalhada da imagem, a partir da imagem de mais baixa resolução, é transmitida. Após a transmissão de todos os coeficientes detalhe, o receptor terá uma cópia completa da imagem. Este tipo de transmissão é conhecido como transmissão progressiva (*progressive transmission*). Para a transmissão progressiva os coeficientes *wavelet* precisam ser arranjados em ordem de importância. A decomposição em multiresolução da transformada torna-se ideal para isso.

Figura 3.19 – Imagem Lenna em várias resoluções.

Uma visualização da reconstrução gradual da imagem Lenna é apresentado na figura 3.20. Inicialmente a imagem é reconstruída a partir dos coeficientes aproximação do quarto nível de decomposição da imagem (matriz 16 × 16), em seguida com os coeficientes

detalhe do quarto nível de decomposição é possível visualizar a imagem (b). Na seqüência, imagem (c), temos a imagem reconstruída a partir dos coeficientes detalhe do terceiro nível de decomposição e na imagem (d) a reconstrução a partir do segundo nível de decomposição. Finalmente, na imagem (e), temos a imagem original reconstruída.

(a) (b) (c) (d) (e)

Figura 3.20 – Reconstrução progressiva a partir da imagem com menor resolução.

As imagens geralmente apresentam mais informações de baixa freqüência (pouca variação) do que informações de alta freqüência (muita variação). Em virtude disso, muitos dos valores resultantes da aplicação do filtro passa alta (coeficientes detalhe) são muito pequenos em magnitude, como pode ser observado analisando o histograma da imagem Lenna após a aplicação da transformada *wavelet* com oito níveis de decomposição (figura 3.21). A transformada *wavelet* concentra as informações da imagem em um número relativamente pequeno de coeficientes. O histograma apresenta somente um grande pico na origem. Isto significa que muitos coeficientes *wavelets* são iguais a zero e que uma grande quantidade desses coeficientes são próximos de zero. Esta característica é que faz a transformada *wavelet* ideal para compressão de imagens. Os poucos coeficientes *wavelet* diferentes de zero permitem uma boa aproximação da imagem.

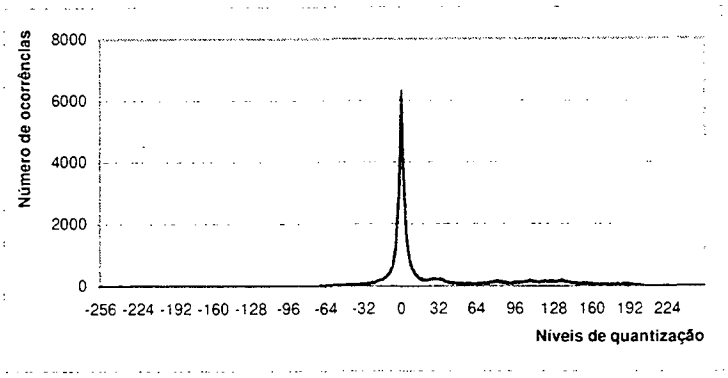


Figura 3.21 – Histograma da imagem da Lenna após a aplicação da transformada *wavelet*.

Os métodos de compressão de imagens baseados na transformada *wavelet* devem tirar vantagem dessa característica das imagens. Armazenando os coeficientes da transformada possibilitará uma redução significante das informações da imagem, sendo que a imagem original não permitiria quase nenhuma compressão. Uma representação gráfica do histograma da imagem original da Lenna de tamanho 256 × 256 e 8 bits por *pixel* e mostrado na figura abaixo.

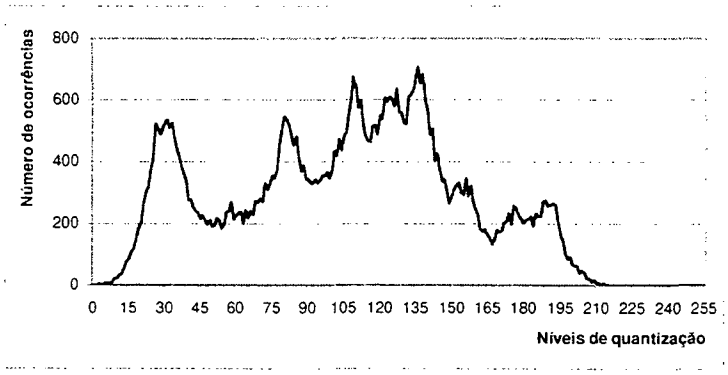


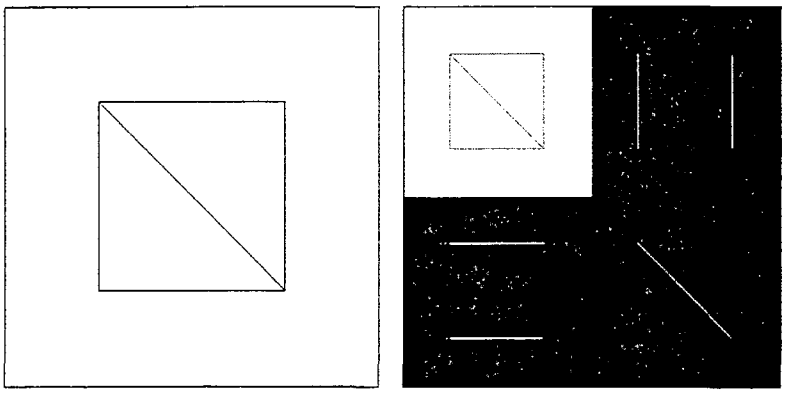
Figura 3.22 – Histograma da imagem original da Lenna.

Se fixarmos um limiar δ e substituir por zero todos os coeficientes *wavelets* da imagem transformada, os quais o valor absoluto é menor ou igual a δ, teremos uma matriz esparsa. Quanto maior for o limiar maior será o fator de compressão, pois uma maior quantidade de valores serão zerados, porém com uma menor qualidade da imagem reconstruída. Este processo é chamado de compressão sem perdas quando nenhuma

informação é perdida, isto é, $\delta = 0$. Caso contrário é chamado de compressão com perdas ($\delta > 0$). Para os casos em que $\delta > 0$ somente uma aproximação da imagem original será possível obter após a reconstrução.

Podemos observar que a quantidade de informação se mantêm a mesma. Para comprimir estes coeficientes devemos aplicar técnicas de codificação sem perdas, tais como: RLE, codificação de Huffman ou codificação aritmética.

Recorrendo a outras técnicas de codificação, tais como LZW, EZW e Predição, podemos ainda explorar a redundância que existe entre os coeficientes das diversas subimagens, podendo aumentar ainda mais o fator de compressão.



(a) (b)

Figura 3.23 – (a) Imagem original. (b) Representação *wavelet* com um nível de resolução.

Como vimos anteriormente, a aplicação da transformada *wavelet* divide a imagem em quatro novas subimagens, (LL, HL, LH e HH). As subimagens detalhes (HL, LH e HH) enfatizam, respectivamente, as características vertical, horizontal e diagonal da imagem, conforme pode ser observado na figura 3.23.

O processo padrão para a realização da compressão (figura 2.1) constitui na aplicação da DWT, quantizar os coeficientes *wavelets* resultantes (para compressão sem perdas os coeficientes não são quantizados) e codificação sem perdas dos coeficientes quantizados. O método mais simples para codificar esses coeficientes é percorrer cada uma

das linhas (da imagem bidimensional), partindo do canto superior esquerdo e finalizando no canto inferior direito. Esta ordem é conhecida como *raster scan*. O problema com o *raster scan* é que as informações relativas as correlações na vertical são perdidas e apresenta uma descontinuidade ao passar de uma linha para outra.

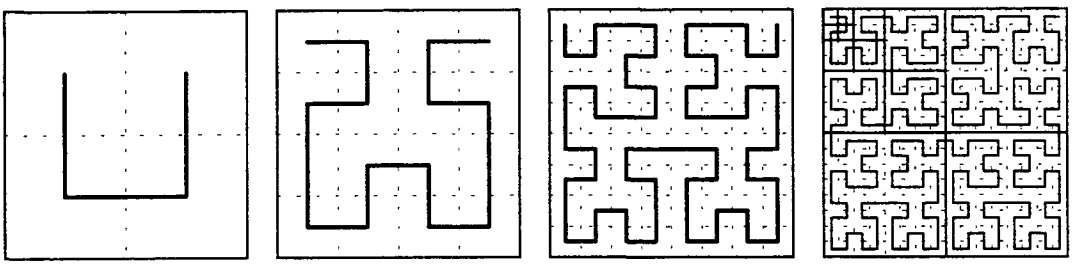


Figura 3.24 – Curva de Hilbert percorrendo um espaço de dimensão 16×16.

Uma outra técnica de codificação, que pode ser aplicada na imagem transformada, é a curva de Hilbert [40, 58]. A Curva de Hilbert permite percorrer a imagem seguindo uma trajetória alternativa e tomar vantagem da correlação dos *pixels* na horizontal e vertical, de cada um dos quadrantes da imagem separadamente.

A curva de Hilbert quebra a direcionalidade dos métodos convencionais para o caso bidimensional, conforme ilustrado na figura 3.24. Uma propriedade desta curva é a divisão dos espaços dentro de quadrantes. Somente após um dos quadrantes ter sido totalmente percorrido, é que o próximo deverá ser percorrido, e assim sucessivamente até a imagem ter sido completamente percorrida.

Um esquema alternativo para codificação dos coeficientes *wavelets* foi recentemente proposto por Shapiro [55] e aprimorado por Said e Pearlman [50], conforme apresentado na seção 1.3.

3.8.2 Compressão de Imagens Coloridas

O processo de compressão de imagens coloridas utilizando a transformada *wavelet* se divide em duas etapas [40]: uma sendo a conversão da imagem do sistema RGB para o sistema YC_bC_r (Y é o canal de luminância e C_bC_r são os dois canais de crominância correspondente ao azul e ao vermelho – também chamado YUV) e outra associada as operações de decomposição, enumeração, quantização e codificação de cada um dos canais.

Como neste trabalho estamos interessados na compressão de imagens com níveis de cinza não iremos detalhar as técnicas usadas para compressão deste tipo de imagem.

4. Transformada *Wavelet* Reversível

4.1 Introdução

A transformada *wavelet* tem sido amplamente utilizada como uma ferramenta poderosa para compressão de imagens. Todavia, até recentemente, seu uso era limitado para aplicações de compressão de imagens com perdas.

Recentemente, a transformada *wavelet* de inteiros (*integer wavelet transform* – IWT), isto é, a transformada *wavelet* que mapeia valores inteiros para inteiros mantendo perfeita reconstrução do sinal de entrada, foi introduzida [2, 4, 5, 12, 16, 51, 52, 56, 71, 72]. Com o objetivo de manter a transformada *wavelet* reversível, temos que nos preocupar com alguns tipos de erros que podem ocorrer durante a aplicação da transformada.

Os métodos clássicos de compressão usando *wavelets* não permitem reconstruir exatamente a versão original da imagem, mesmo retendo todos os coeficientes resultantes da aplicação da transformada *wavelet*. Isso ocorre pelo fato de que os filtros utilizados na transformada *wavelet* geram coeficientes que são números reais (números com ponto flutuante) e têm que ser convertidos para números inteiros. Em muitos casos a aplicação da transformada, no sentido matemático, permite a reconstrução perfeita dos dados originais, porém na prática as operações de truncamento e/ou o armazenamento dos coeficientes com aritmética de precisão finita é que provocam a perda de informação e impede a sua exata recuperação quando se aplica a transformada inversa. Este fato é uma limitação para compressão de imagens sem perdas através dos algoritmos baseados na transformada *wavelet*.

Os dados de uma imagem, na sua forma matricial, são constituídos de valores inteiros e a saída filtrada deverá consistir de valores inteiros também, por isso, perdas

ocorrerão por causa do arredondamento dos valores reais gerados durante a filtragem. Para codificação sem perdas é necessário então aplicar uma transformada *wavelet* reversível que permita mapear os dados de entrada da imagem de inteiros para inteiros. Neste caso, as operações aritméticas são efetuadas utilizando números de ponto flutuante, porém o resultado será um valor inteiro e a transformada continuará sendo reversível.

Tendo em vista as limitações do computador ao efetuar o armazenamento de números em aritmética de ponto flutuante, temos que nos preocupar com os filtros a serem utilizados e também com a forma que será efetuada o truncamento. Para resolver este problema, devemos arredondar o valor dos coeficientes para o menor inteiro (função "*floor*") ou para o maior inteiro (função "*ceil*"). Não podemos apenas aplicar um truncamento, pois os valores negativos seriam truncados para cima e não para o menor valor inteiro.

O número máximo de bits requerido para representar cada coeficiente aproximação não muda com a aplicação da transformada *wavelet*. Por outro lado, os coeficientes detalhe podem assumir valores negativos, requerendo uma representação sinalizada o que exigirá um número maior de bits. Por exemplo, se assumirmos uma imagem com oito bits, dependendo do filtro, após a aplicação da transformada serão necessários nove bits ou mais para armazenar cada coeficiente. Uma maneira de contornar este problema é mantendo o valor dos coeficientes num intervalo entre $[-2^{q-1}, 2^{q-1} - 1]$, para q igual ao número de bits necessários para representar cada *pixel* da imagem (por ex. 1, 8, 16, ...), conforme mostrado em [3, 6, 7].

Além de evitar perdas no processo de compressão, a transformada *wavelet* de inteiros permite realizar a computação de forma mais rápida do que com número de ponto flutuante. A computação de inteiros também se torna mais fácil para implementação em hardware e a quantidade de memória necessária nos cálculos com inteiros também é um ponto positivo. Quando se trata de compressão com perdas, o uso de computação de inteiros

pode se tornar mais vantajoso, em termos de velocidade de processamento, do que usar aritmética de ponto flutuante.

4.2 Transformadas Wavelet de Inteiros Reversíveis

Wavelets reversíveis são filtros lineares com arredondamento não linear os quais implementa sistemas de reconstrução exata utilizando aritmética de inteiros [72]. As transformadas reversíveis, em geral, não são lineares. Por este motivo, a ordem na qual a transformada é aplicada torna-se significante, ou seja, se a decomposição foi aplicada primeiro nas linhas e em seguida nas colunas da imagem, a transformada inversa deve primeiro ser aplicada nas colunas e então nas linhas da imagem.

Como exemplo, apresentaremos a seguir três transformadas reversíveis clássicas já citadas anteriormente. Consideremos, $x[n]$, $n = 0, 1, ..., N-1$, um sinal de entrada de tamanho N, $N = 2^p$, $p = 1, 2, ...$; $s[n]$ e $d[n]$ representam os coeficientes aproximação e detalhe de saída respectivamente. Os coeficientes de saída são resultados da aplicação dos filtros passa baixa e passa alta, respectivamente, no sinal de entrada.

4.2.1 Transformada S

A transformada *wavelet* de Haar na sua versão não normalizada envolve simplesmente pares de médias e diferenças: [4]

$$s[n] = \frac{x[2n]+x[2n+1]}{2}$$
$$d[n] = x[2n+1]-x[2n]. \qquad (4.1)$$

Para o caso de uma imagem, devemos aplicar a transformada acima nas linhas e nas colunas da imagem, conforme mostrado na seção 3.8.1. A transformada inversa de Haar é dada por

$$x[2n] = s[n] - \frac{d[n]}{2}$$
$$x[2n+1] = s[n] + \frac{d[n]}{2}. \qquad (4.2)$$

Por causa da divisão por dois, as equações acima não são transformadas inteiras. Para evitar isso, poderíamos omitir a divisão por dois calculando a soma em vez da média, conforme proposto em [4]. Porém, é preferível usar uma construção mais eficiente conhecida como transformada S (Sequencial) [2, 4, 5, 51, 52, 53, 72] que é uma versão da transformada de Haar modificada para inteiros. A transformada S é a mais antiga transformada *wavelet* de inteiro para inteiro. Existe na literatura diferentes definições da transformada S, mas muitas diferem apenas na forma em que são implementadas. A transformada S será definida aqui como segue:

$$s[n] = \left\lfloor \frac{x[2n] + x[2n+1]}{2} \right\rfloor$$
$$d[n] = x[2n+1] - x[2n] \qquad (4.3)$$

onde $\lfloor \cdot \rfloor$ corresponde a operação de truncamento para baixo. A primeira vista, o truncamento em $s[n]$ parece descartar alguma informação. Todavia, a soma e a diferença de dois números inteiros são também ambos pares ou ambos impares. Com base nessa informação, podemos omitir o bit menos significativo da soma, pois ele é igual ao bit menos significativo da diferença ($d[n]$). A diferença poderá então ser usada para determinar se ocorreu o truncamento. Se este valor for impar indicará o truncamento para o menor inteiro.

Note que o índice de fator dois ($[2n]$), nos coeficientes da transformada, é o resultado da subamostragem por um fator de 2, enquanto que a operação de truncamento $\lfloor \cdot \rfloor$ é a fonte da não linearidade.

Como os *pixels* adjacentes em uma imagem são altamente correlacionados, a aplicação da transformada acima permite reduzir significantemente a entropia de primeira ordem da imagem, que pode ser reduzida ainda mais se for usado um método de codificação preditiva.

A transformada S é invertível e sua inversa é dada por

$$\begin{aligned} x[2n] &= s[n] - \left\lfloor \frac{d[n]}{2} \right\rfloor \\ x[2n+1] &= s[n] + \left\lfloor \frac{d[n]+1}{2} \right\rfloor. \end{aligned} \tag{4.4}$$

4.2.2 Transformada S+P

Said e Pearlman [4, 51, 52] propuseram a transformada S+P (Transformada S + Predição) na qual a predição linear é realizada nos coeficientes passa baixa para gerar um novo conjunto de coeficientes passa alta depois da aplicação da transformada S. A forma geral da transformada S+P é:

$$d^{(1)}[n] = x[2n+1] - x[2n] \tag{4.5}$$

$$s[n] = x[2n] + \left\lfloor \frac{d^{(1)}[n]}{2} \right\rfloor \tag{4.6}$$

$$d[n] = d^{(1)}[n] + \left\lfloor \begin{matrix} \alpha_{-1}(s[n-2] - s[n-1]) + \alpha_0(s[n-1] - s[n]) \\ + \alpha_1(s[n] - s[n+1]) - \beta_1 d^{(1)}[n+1] \end{matrix} \right\rfloor \tag{4.7}$$

Said e Pearlman examinaram várias escolhas para α_n e β_1 e sugerem $\alpha_{-1} = 0$, $\alpha_0 = \frac{2}{8}$, $\alpha_1 = \frac{3}{8}$ e $\beta_1 = \frac{2}{8}$ para imagens naturais e $\alpha_{-1} = -\frac{1}{16}$, $\alpha_0 = \frac{4}{16}$, $\alpha_1 = \frac{8}{16}$ e $\beta_1 = \frac{6}{16}$ para imagens médicas muito suave.

4.2.3 Transformada TS

Similar a transformada S, a transformada TS (*TS-Transform* ou *Two-Six Transform*, devido ao número de derivações (*taps*) nos filtros passa baixa e passa alta respectivamente) é definida por: [53, 72]

$$s[n] = \frac{x[2n] + x[2n+1]}{2} \tag{4.8}$$

$$d[n] = \left\lfloor \frac{-\left\lfloor \frac{x[2n] + x[2n+1]}{2} \right\rfloor + 4x[2n+2] - 4x[2n+3] + \left\lfloor \frac{x[2n+4] + x[2n+5]}{2} \right\rfloor}{4} \right\rfloor \tag{4.9}$$

A expressão para $d[n]$ pode ser simplificada e escrita com o uso de $s[n]$. Isto resulta em:

$$d[n] = x[2n+2] - x[2n+3] + \left\lfloor \frac{-s[n] + s[n+2] + 2}{4} \right\rfloor \tag{4.10}$$

A transformada TS é reversível e a sua inversa é dada por:

$$\begin{aligned} x[2n] &= s[n] + \left\lfloor \frac{p[n]+1}{2} \right\rfloor \\ x[2n+1] &= s[n] - \left\lfloor \frac{p[n]}{2} \right\rfloor \end{aligned} \tag{4.11}$$

onde $p[n]$ deve ser primeiro computado por

$$p[n] = d[n-1] - \left\lfloor \frac{-s[n-1] + s[n+1] + 2}{4} \right\rfloor \tag{4.12}$$

A transformada TS é um caso especial da transformada S+P quando $\alpha_{-1} = \beta_1 = 0$ e $\alpha_0 = \alpha_1 = \frac{1}{4}$. As transformadas S, S+P e TS podem ser vistas como casos especiais da transformada *wavelet* que mapeia inteiros para inteiros [4] baseado na transformada em termos de *lifting*.

4.3 Wavelets Biortogonais

A propriedade de ortogonalidade coloca limitações na construção de *wavelets*. Por exemplo, a transformada *wavelet* de Haar é a única *wavelet* de valor real que é suportada compactamente, simétrica e ortogonal. A grande vantagem de *wavelets* biortogonais é que elas podem ser simétricas, enquanto que as *wavelets* ortonormais não podem, exceto a *wavelet* de Haar. A generalização para *wavelets* biortogonais tem sido considerada para ganhar mais flexibilidade [29]. Aqui, existe uma função de escala dual $\tilde{\varphi}$ e uma *wavelet* dual $\tilde{\psi}$ que geram uma análise em multiresolução com subespaços $\tilde{V}_j$ e $\tilde{W}_j$, tais que

$$\tilde{V}_j \perp W_j \quad \text{e} \quad V_j \perp \tilde{W}_j, \tag{4.13}$$

e consequentemente,

$$\tilde{W}_j \perp W_{j'} \qquad \text{para } j \neq j'. \tag{4.14}$$

Os filtros H, G, $\tilde{H}$ e $\tilde{G}$ são chamados filtros *wavelet* se eles preenchem certas condições. Os filtros $\tilde{H}$ e $\tilde{G}$ são chamados de filtros dual em relação a H e G [65]. Os filtros *wavelet* H e G definem unicamente uma função *wavelet* primária (*primal*) $\psi(t)$ e uma função de escala primária $\varphi(t)$. Analogamente, os filtros $\tilde{H}$ e $\tilde{G}$ definem uma função *wavelet* dual $\tilde{\psi}(t)$ e uma função de escala dual $\tilde{\varphi}(t)$.

Se os filtros G e $\tilde{G}$, e H e $\tilde{H}$ são iguais um com o outro, então a *wavelet* primária e dual e as funções de escala coincidem também. Neste caso as *wavelets* são chamadas *wavelets* ortogonais. Em um caso mais genérico, elas são chamadas *wavelets* biortogonais [65].

A função de escala $\varphi(t)$ e a função *wavelet* $\psi(t)$ e suas duais $\tilde{\varphi}(t)$ e $\tilde{\psi}(t)$ satisfazem a equação de dilatação que são:

$$\varphi(t) = \sqrt{2}\sum_{k\in Z} h_k \varphi(2t-k), \tag{4.15}$$

$$\tilde{\varphi}(t) = \sqrt{2}\sum_{k\in Z} \tilde{h}_k \tilde{\varphi}(2t-k), \tag{4.16}$$

$$\psi(t) = \sqrt{2}\sum_{k\in Z} g_k \varphi(2t-k), \tag{4.17}$$

$$\tilde{\psi}(t) = \sqrt{2}\sum_{k\in Z} \tilde{g}_k \tilde{\varphi}(2t-k). \tag{4.18}$$

Regularidade é freqüentemente um requerimento na função de escala $\varphi(t)$ e representa o grau de suavidade da função de escala. Como a função *wavelet* $\psi(t)$ é derivada da função de escala, elas compartilham algumas características comuns [47]. A regularidade de funções bases podem ser medidas pelo número de momentos de anulação (*vanishing moments*). A função tem N momentos de anulação se os primeiros N momentos M_j ($j \leq N$)

$$M_j = \int_{-\infty}^{\infty} x^j f(t)dt, \tag{4.19}$$

são zero. Quanto mais momentos de anulação uma função tem, mais suave é a função. A *wavelet* de Haar tem um momento de anulação, portanto a projeção de uma função aproximadamente constante em uma base de Haar terá coeficientes *wavelet* próximos de zero. Analogamente, se uma base *wavelet* possui dois, a projeção de funções lineares irão desvanecer rapidamente. O número de momentos de anulação determina a taxa de convergência. Onde a imagem é suave mais momentos de anulação produz coeficientes *wavelet* mais pequenos. Por outro lado, onde a imagem não é suave mais momentos de anulação produz coeficientes *wavelet* mais grande. Filtros *wavelet* com mais momentos de anulação geralmente atuam melhor com imagens suaves e naturais e não tão bem com imagens com muitas bordas e componentes de alta freqüência.

As *wavelets* biortogonais são classificadas de acordo com o número de momentos de anulação que elas têm. A notação $(N, \tilde{N})$ significa que a *wavelet* primária $\psi(t)$ - filtro passa alta de análise - tem N momentos de anulação, enquanto que a *wavelet* dual $\tilde{\psi}(t)$ - filtro passa alta de síntese - tem $\tilde{N}$ momentos de anulação.

Seja N o número de momentos de anulação da *wavelet* dual, então [59, 60]

$$\int_{-\infty}^{\infty} x^p \tilde{\psi}(x)\, dx = 0 \quad \text{para} \quad 0 \le p < N. \tag{4.20}$$

Similarmente, as *wavelets* têm $\tilde{N}$ momentos de anulação. Na análise em multiresolução, N e $\tilde{N}$ são pelo menos 1. A *wavelet* de Haar possui um momento de anulação. A ordem do filtro na análise em multiresolução é definido por N, para $N \in \mathbf{N}$, onde as *wavelets* são suportadas num intervalo $2N-1$.

O caso $\tilde{N} \le N$ é o mais comum enquanto que $\tilde{N} > N$ é menos interessante e mais complexo. Em processamento de imagens o número de momentos de anulação dual é muito mais importante do que o número de momentos de anulação primária. A prova pode ser obtida em [60].

Suporte é definido como um intervalo fechado em tempo contínuo, os quais fora deste intervalo a *wavelet* é zero. Suporte compacto significa que o conjunto fechado é limitado, ou seja, somente uma pequena área ao redor do ponto sendo analisado é afetado. A *wavelet* vale zero fora do intervalo limitado. Filtros de resposta finita ao impulso (*finite impulse response* – FIR) são limitados por definição, e portanto, são perfeitamente adaptados para suporte compacto. Na decomposição de um sinal os filtros de resposta infinita ao impulso (*infinite impulse response* – IIR) também possam ser usados. Eles não são vantajosos

porque a resposta infinita conduz para expansão infinita dos dados e para uso prático a saída do banco de filtros IIR levará a perda de dados.

Wavelets possuem suporte local em espaço e freqüência. Localização em espaço segue do suporte compacto delas, enquanto que a localização em freqüência segue da suavidade delas (decai para altas freqüências) e dos momentos de anulação (decais para baixas freqüências). A função *wavelet* $\psi(t)$ é localizada no sentido em que a sua magnitude decaí rapidamente para zero, quando a função tende para mais ou menos infinito. Isto significa que somente uma pequena área ao redor do corrente coeficiente sendo analisado será afetado.

4.4 Lifting Scheme

A idéia básica da transformada *wavelet* é explorar a correlação presente nos sinais para construir uma aproximação esparsa. A correlação é tipicamente local em espaço (tempo) e freqüência. Na construção de *wavelet* tradicional é usado a transformada de Fourier para construir a localização em espaço-freqüência. Todavia, isto pode ser feito no domínio espacial. A principal característica do *lifting scheme* é que todas as construções são derivadas no domínio espacial, contrastando com a abordagem tradicional os quais é no domínio da freqüência.

Esta idéia de usar *wavelet* espacial têm sido proposto por vários pesquisadores [12]. O *lifting scheme* foi inspirado no trabalho de Donoho e do Lounsbery *et al.* [12]. Donoho mostra como construir *wavelets* interpolando funções de escala, enquanto Lounsbery *et al.* constrói uma análise em multiresolução de superfícies usando uma técnica que é algebricamente o mesmo do *lifting*. Além desses, outros trabalhos foram desenvolvidos independentemente.

O termo "*lifting*" foi criado por Sweldens onde ele foi desenvolvido para elevar ("*lift*") o número de momentos de anulação de uma *wavelet* [5]. O *lifting scheme* é um algoritmo originalmente projetado para computar *wavelets* da segunda geração de uma maneira eficiente [12, 30, 59, 60, 61, 62, 64, 65]. O *lifting scheme* é um novo método para construir *wavelets* biortogonais com suporte compacto [62]. A principal diferença com a construção clássica é que ele não é introduzido usando a transformada de Fourier. Desta maneira *lifting* pode ser usado para construir *wavelets* da segunda geração, os quais não são necessariamente translações e dilatações de uma determinada função, mantendo todavia, as propriedades das *wavelets* da primeira geração.

Este esquema pode ser usado para gerar as *wavelets* (bi)ortogonais de Cohen-Daubechies-Feaveau (CDF) [9] com várias vantagens em relação aos esquemas computacionais clássicos. Em [12] é mostrado como qualquer transformada *wavelet* discreta com filtros finitos pode ser decomposto em uma seqüência finita com simples passos de filtragem, os quais são chamados de passos *lifting* mas que são também conhecidos como estruturas escada (*ladder structures*). É mostrado também que em quase todas elas a complexidade computacional pode ser reduzida até pela metade quando comparado com os algoritmos padrões.

O *lifting scheme* possui as seguintes vantagens: [62]

- método genérico.
- ele permite uma implementação mais rápida da transformada *wavelet* ($O(n)$). O *lifting scheme* faz uso eficiente das similaridades entre os filtros passa alta e passa baixa para aumentar a velocidade dos cálculos. O número necessário de operações de ponto flutuante podem ser reduzidos pela metade.

- o *lifting scheme* permite sobrescrever os dados originais pelos coeficientes da transformada *wavelet* na mesma localização de memória (*in-place*). Isto significa que a transformada *wavelet* pode ser calculada sem alocação de memória auxiliar.
- a transformada *wavelet* inversa pode ser imediatamente encontrada através do processo inverso da transformada direta. Na prática, isto é equivalente a mudança de sinal na fórmula de + (adição) para – (subtração) e vice versa. A transformada inversa possui a mesma complexidade da transformada direta.
- fácil de entender, já que não é introduzido usando a transformada de Fourier.
- todos os filtros *wavelets* podem ser implementados usando *lifting scheme*.
- transforma um sinal com um tamanho arbitrário, não necessitando possuir tamanho da forma 2^n.
- É particularmente fácil de construir transformadas *wavelet* não linear. Um exemplo típico são as transformadas *wavelet* que mapeiam inteiros para inteiros. Estas transformadas são importantes para codificação de imagens sem perdas.

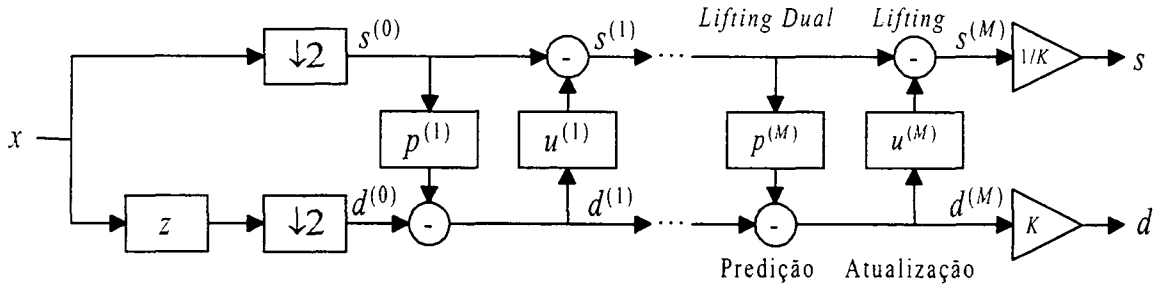


Figura 4.1 – A transformada *wavelet* direta usando *lifting*.

A figura 4.1 representa um esquema genérico do *lifting scheme* [2, 5, 12, 64, 65] para a transformada direta. Neste esquema, uma transformada *wavelet* trivial, chamada transformada *wavelet* "preguiçosa" (*lazy wavelet transform*) é calculado primeiro. Esta

transformada é uma transformada ortogonal que essencialmente não faz nada. Ela tem apenas a função de dividir o sinal de entrada x em duas seqüências contendo em uma delas as amostras pares e na outra as amostras ímpares do sinal, os quais são chamados componentes polifásico. Dessa maneira, teremos as amostras pares e ímpares dadas por $s_j^{(0)} = x_{2j}$ e $d_j^{(0)} = x_{2j+1}$, para $j \in \mathbf{Z}$, respectivamente.

Em seguida, a transformada é elevada (*lifted*) para uma transformada com as propriedades desejadas usando um ou mais operações dos filtros p e/ou u. Primeiro calcula-se os coeficientes *wavelets* (d) e em seguida usa eles para elevar os coeficientes aproximação (s). Por fim, os resultados são normalizados multiplicando-os pelos fatores $\frac{1}{K}$ e K respectivamente. Para a obtenção da transformada inversa basta inverter a ordem dos passos acima.

Os passos *lifting* dual também são referidos como passos de predição (*predict step*) enquanto que os passos *lifting* são também referidos como passos de atualização (*update step*).

A transformada de Haar usa um preditor que é correto no caso em que o sinal original é uma constante. Ela elimina a correlação de ordem zero. Nesse caso dizemos que a ordem do preditor é um. Similarmente, a ordem do operador de atualização é um quando ele preserva a média ou o momento de ordem zero. Para o caso em que temos uma predição e uma atualização os quais a ordem é dois, o preditor será exato no caso em que o sinal original é linear e a atualização preservará a média e o momento de ordem um [10]. Em geral, *wavelets* de maior ordem tendem a colocar mais informações nas saídas ímpares e menos nas saídas pares.

O resultado da transformada *wavelet* usando o *lifting scheme* conterá todos os coeficientes em uma forma intercalada. Todavia, a ordem clássica de Mallat [32] pode ser utilizada dependendo da aplicação.

4.4.1 Algoritmo

Apresentaremos a seguir o fluxo do algoritmo da transformada *wavelet* direta, mostrado graficamente na figura 4.1, usando o *lifting scheme* [2, 5]:

1. Dividir (*Split*): Seja $x[n]$ o sinal de entrada, a transformada *wavelet* preguiçosa é dada por

$$s^{(0)}[n] \leftarrow x[2n]$$

$$d^{(0)}[n] \leftarrow x[2n+1]$$

2. Passos *lifting* (*lifting steps*): Um ou mais passos M da forma

   a. *Lifting* dual: Consiste em aplicar um filtro para as amostras pares e subtrair o resultado das amostras ímpares

$$d^{(i)}[n] = d^{(i-1)}[n] - \sum_k p^{(i)}[k] s^{(i-1)}[n-k]$$

   b. *Lifting*: Consiste em aplicar um filtro para as amostras ímpares e subtrair o resultado das amostras pares

$$s^{(i)}[n] = s^{(i-1)}[n] - \sum_k u^{(i)}[k] d^{(i)}[n-k]$$

3. Normalização: Finalmente, as amostras pares $s^{(M)}[n]$ tornam-se os coeficientes passa baixa $s[n]$ e as amostras ímpares $d^{(M)}[n]$ tornam-se os coeficientes passa alta $d[n]$ quando escalonados por um fator K

$$s[n] = \frac{s^{(M)}[n]}{K}$$

$$d[n] = K d^{(M)}[n]$$

Como pode ser observado, a transformada inversa é calculada invertendo os passos do algoritmo e os sinais de subtração por sinais de adição.

Quando aplicamos um ou mais passos *lifting* primário e/ou dual com a transformada preguiçosa conseguimos uma nova transformada *wavelet* que é um pouco mais sofisticada. Em outras palavras, elevamos (*lifted*) a transformada *wavelet* para um nível mais alto de sofisticação. Podemos realizar tantos passos *lifting* quanto quisermos e dessa maneira construir transformadas *wavelet* altamente sofisticadas.

4.4.2 Transformada *Wavelet* de Inteiros usando *Lifting*

O algoritmo apresentado anteriormente, em geral, provocam perdas de informações. Entretanto, o *lifting scheme* pode ser facilmente modificado para construir *wavelets* reversíveis os quais mapeia inteiros para inteiros e permite perfeita reconstrução em aplicações de compressão de imagens sem perdas. Para isso, será adicionado uma operação de arredondamento, introduzindo uma não linearidade na transformada.

Um passo *lifting* se parece basicamente com

$$x[n] \leftarrow x[n] - \frac{1}{a}\sum_{m} b[m]y[m] \qquad (4.21)$$

onde a, $b[m] \in \mathbf{Z}$. Isto é verdade para várias transformadas *wavelet*, entre elas as *wavelets* biortogonais de Cohen-Daubechies-Feaveau [9].

Este passo *lifting* pode ser modificado por uma das maneiras seguintes com objetivo de manter os valores inteiros [64, 65]. O valor $\{x\}$ representa o arredondamento inteiro de x. Esta função pode ter diferentes interpretações, ou seja, $\{x\}$ pode ser o valor inteiro mais próximo de x, ou $\{x\}$ pode ser qualquer valor inteiro os quais satisfaz $x - 1 < \{x\} \leq x$, etc. Geralmente usa-se o arredondamento para baixo ("*floor*").

- Arredondamento completo – o resultado da divisão por a é arredondado:

$$\tilde{x}[n] \leftarrow x[n] - \left\{ \frac{\sum_m b[m]y[m]}{a} \right\} \qquad (4.22)$$

onde $\tilde{x}[n]$ representa uma aproximação inteira para $x[n]$.

- Sem arredondamento – evita-se a divisão por a multiplicando os outros termos por a:

$$\tilde{x}[n \leftarrow ax[n] - \sum_m b[m]y[m]. \qquad (4.23)$$

Aqui $\tilde{x}[n] = ax$, e desse modo o intervalo dinâmico dos coeficientes *wavelet* aumentará. Nesse caso nenhum arredondamento real é realizado.

- Forma mista – combina os passos de arredondamento e multiplicação de ambos os métodos anteriores:

$$\tilde{x}[n] \leftarrow a_1 x[n] - \left\{ \frac{\sum_m b[m]y[m]}{a_2} \right\} \qquad (4.24)$$

com $a_1, a_2 \in \mathbf{Z}$ e $a_1.a_2 = a$.

Note que em todos os três casos anteriores os passos *lifting* modificados permanecem reversíveis, e dessa maneira a reconstrução exata ainda mantêm presente.

Como exemplo, consideremos a transformada S (transformada de Haar modificada) implementada usando *lifting scheme* (CDF(1, 1)). A versão inteira é calculado da seguinte maneira [64, 65]:

Transformada Direta		**Transformada Inversa**	
Divisão:	$s[n] \leftarrow x[2n]$ $d[n] \leftarrow x[2n+1]$	*Lifting* primário:	$s[n] \leftarrow s[n] - \left\{ \frac{d[n]}{2} \right\}$
Lifting dual:	$d[n] \leftarrow d[n] - s[n]$	*Lifting* dual:	$d[n] \leftarrow d[n] + s[n]$
Lifting primário:	$s[n] \leftarrow s[n] + \left\{ \frac{d[n]}{2} \right\}$	União:	$x[2n] \leftarrow s[n]$ $x[2n+1] \leftarrow d[n]$

No exemplo acima consideramos o fator de normalização K = 1.

Podemos observar que a transformada direta, do exemplo acima, é o mesmo da transformada S da equação 4.3 pois, $s[n]+\left\{\frac{d[n]}{2}\right\} = s[n]+\left\{\frac{d[n]-s[n]}{2}\right\} = \left\{\frac{s[n]+d[n]}{2}\right\} = \left\{\frac{x[2n]+x[2n+1]}{2}\right\}$.

Uma representação gráfica da transformada direta, do exemplo anterior, é mostrado na figura abaixo.

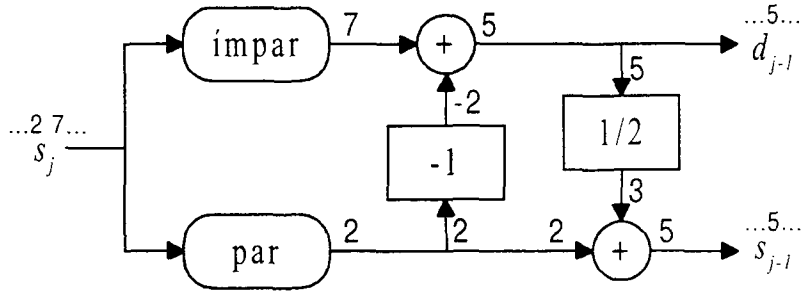


Figura 4.2 – Representação gráfica da transformada *wavelet* de Haar modificada.

A multiplicação dos coeficientes pelos fatores de normalização não resultará em perdas de informação. A transformada *wavelet* bidimensional é sempre aplicada na direção vertical e horizontal da matriz. Uma vez que $\sqrt{2}.\sqrt{2} = 2$, os fatores de normalização sempre resultarão em valores racionais.

Apresentaremos a seguir as transformadas *wavelet* de inteiros utilizadas na compressão das projeções neste trabalho. Elas são baseadas nas transformadas citadas por [6, 7, 47], os quais possuem a propriedade de preservação de precisão (*property of precision preservation* – PPP) e permitem a compressão sem perdas de imagens com tamanho arbitrário. Estas transformadas são baseadas em técnicas de atualização e correção, tais como o *lifting scheme* e a transformada S+P. O método permite gerar uma série de transformações inteiras reversíveis que são muito próximas das transformadas *wavelet* biortogonais

correspondentes e algumas transformadas *wavelet* não ortogonais, porém podem ser calculadas com somas de inteiros e operações de deslocamento de bits.

Como explicado anteriormente, após a aplicação da transformada direta os coeficientes detalhe podem assumir valores negativos e o número de bits usados para representar a imagem podem não ser mais suficientes, havendo a necessidade de se usar um bit adicional para representar o sinal do número. A PPP permite manter o número de bits usados para representar cada *pixel* da imagem inalterado. No algoritmo de reconstrução o computador recuperará o sinal original através da mesma propriedade.

Alguns dos coeficientes obtidos da transformada PPP terão valores de saída diferentes dos obtidos com a transformada sem a aplicação desta propriedade. Em [7] é apresentado uma comparação entre a transformada PPP e a mesma transformada sem a utilização desta propriedade. Os resultados obtidos com o uso desta propriedade possuem aproximadamente a mesma eficiência na compressão sem perdas, que significa que a entropia da imagem transformada dessas duas transformadas são quase a mesma. A transformada PPP não é aplicável para compressão com perdas.

Uma outra característica da PPP está na forma como o computador realiza aritmética com números inteiros. Se considerarmos a computação da diferença de dois números inteiros dado como $c = b - a$ e a computação inversa de $a = b - c$. A computação será efetuado da seguinte maneira pelo computador [6, 7]:

$$c' = \begin{cases} b-a & se\ -2^{q-1} \le b-a < 2^{q-1} \\ b-a-2^{q} & se\ b-a \ge 2^{q-1} \\ b-a+2^{q} & se\ b-a < -2^{q-1} \end{cases} \qquad (4.25)$$

e a inversa será

$$a' = \begin{cases} b-c' & se\ -2^{q-1} \le b-c' < 2^{q-1} \\ b-c'-2^{q} & se\ b-c' \ge 2^{q-1} \\ b-c'+2^{q} & se\ b-c' < -2^{q-1} \end{cases} \qquad (4.26)$$

onde c' e a' indicam a representação interna no computador e o intervalo dos inteiros a, b e c é $[-2^{q-1}, 2^{q-1}-1]$. A representação interna de c' quando ocorrer um *overflow* ou *underflow* será um número com complemento de dois, então a representação interna pode não ser a mesma da representação externa de c. Porém, o mesmo código complementar para a' permitirá que a representação interna seja igual a representação externa de a.

4.4.2.1 Transformada S (1, 1)

Esta transformada é a mesma do exemplo apresentado na seção 4.4.2 usando *lifting scheme*. Porém, devido a natureza do código complementar e a computação realizada na máquina, os cálculos serão automaticamente realizados pela máquina como segue:

1. Decomposição:

   - Dividir: $s[n] = x[2n]$
     $d[n] = x[2n+1]$

   - Computar:

$$d[n] = \begin{cases} d[n] - s[n] & se \ -2^{q-1} \le d[n] - s[n] < 2^{q-1} \\ d[n] - s[n] - 2^q & se \ d[n] - s[n] \ge 2^{q-1} \\ d[n] - s[n] + 2^q & se \ d[n] - s[n] < -2^{q-1} \end{cases}$$

$$s[n] = \begin{cases} s[n] + \left\{\frac{d[n]}{2}\right\} & se \ -2^{q-1} \le s[n] + \left\{\frac{d[n]}{2}\right\} < 2^{q-1} \\ s[n] + \left\{\frac{d[n]}{2}\right\} - 2^q & se \ s[n] + \left\{\frac{d[n]}{2}\right\} \ge 2^{q-1} \\ s[n] + \left\{\frac{d[n]}{2}\right\} + 2^q & se \ s[n] + \left\{\frac{d[n]}{2}\right\} < -2^{q-1} \end{cases}$$

2. Reconstrução

   - Computar:

$$s[n] = \begin{cases} s[n] - \left\{\frac{d[n]}{2}\right\} & se \ -2^{q-1} \le s[n] - \left\{\frac{d[n]}{2}\right\} < 2^{q-1} \\ s[n] - \left\{\frac{d[n]}{2}\right\} - 2^q & se \ s[n] - \left\{\frac{d[n]}{2}\right\} \ge 2^{q-1} \\ s[n] - \left\{\frac{d[n]}{2}\right\} + 2^q & se \ s[n] - \left\{\frac{d[n]}{2}\right\} < -2^{q-1} \end{cases}$$

$$d[n] = \begin{cases} d[n]+s[n] & se\ -2^{q-1} \le d[n]+s[n] < 2^{q-1} \\ d[n]+s[n]-2^{q} & se\ d[n]+s[n] \ge 2^{q-1} \\ d[n]+s[n]+2^{q} & se\ d[n]+s[n] < -2^{q-1} \end{cases}$$

- Unir: $x[2n] = s[n]$
  $x[2n+1] = d[n]$

4.4.2.2 Transformada (2, 6)

Esta transformada possui os seguintes filtros de análise passa baixa e passa alta:

n	-2	-1	0	1	2	3
$\tilde{H}$	0	0	$\frac{1}{2}$	$\frac{1}{2}$	0	0
$\tilde{G}$	$\frac{-1}{16}$	$\frac{-1}{16}$	$\frac{1}{2}$	$\frac{-1}{2}$	$\frac{1}{16}$	$\frac{1}{16}$

usando o fator de escala igual a 1 para a parte passa baixa e 2 para a parte passa alta.

A decomposição inteira usando os filtros acima inicia com a transformada S e então atualiza os componentes de alta freqüência no passo seguinte.

1. Decomposição:

   - Dividir: $s[n] = x[2n]$
     $d[n] = x[2n+1]$

   - Computar a transformada S:

$$d[n] = d[n] - s[n]$$

$$s[n] = s[n] + \left\{ \frac{d[n]}{2} \right\}$$

   - Atualizar os componentes de alta freqüência:

$$d[n] = d[n] - \left\{ \frac{s[n-1] - s[n+1]}{4} \right\}$$

2. Reconstrução

   - Computar:

$$d[n] = d[n] + \left\{ \frac{s[n-1] - s[n+1]}{4} \right\}$$

$$s[n] = s[n] - \left\{ \frac{d[n]}{2} \right\}$$

$$d[n] = d[n] + s[n]$$

- Unir:
$$x[2n] = s[n]$$
$$x[2n+1] = d[n]$$

4.4.2.3 Transformada (1, 3)

Esta transformada é uma variação da transformada que usa os seguintes filtros de análise biortogonais:

n	**-1**	**0**	**1**
$\tilde{H}$	1	0	0
$\tilde{G}$	$\frac{1}{4}$	$\frac{-1}{2}$	$\frac{1}{4}$

A decomposição inteira dos filtros acima inicia com a transformada *wavelet* preguiçosa (*lazy*) e então atualiza os componentes de alta freqüência no passo seguinte.

1. Decomposição:

   - Dividir:
   $$s[n] = x[2n]$$
   $$d[n] = x[2n+1]$$

   - Atualizar os componentes de alta freqüência:

   $$d[n] = \left\{ \frac{s[n] + s[n+1]}{2} \right\} - d[n]$$

2. Reconstrução

   - Computar:

   $$d[n] = \left\{ \frac{s[n] + s[n+1]}{2} \right\} - d[n]$$

   - Unir:
   $$x[2n] = s[n]$$
   $$x[2n+1] = d[n]$$

4.4.2.4 Transformada (5, 3)

Esta transforma também usa os filtros de análise biortogonal dado por:

n	-2	-1	0	1	2
$\widetilde{H}$	$\frac{-1}{8}$	$\frac{1}{4}$	$\frac{3}{4}$	$\frac{1}{4}$	$\frac{-1}{8}$
$\widetilde{G}$	0	0	$\frac{-1}{4}$	$\frac{1}{2}$	$\frac{-1}{4}$

Será usado os fatores de escala da transformada (2, 6). A decomposição inteira dos filtros acima inicia com a transformada *wavelet* preguiçosa (*lazy*) e então atualiza os componentes de alta e de baixa freqüência respectivamente.

1. Decomposição:

   - Dividir:
   $$s[n] = x[2n]$$
   $$d[n] = x[2n+1]$$

   - Atualizar os componentes de alta freqüência:
   $$d[n] = \left\{\frac{s[n] + s[n+1]}{2}\right\} - d[n]$$

   - Atualizar os componentes de baixa freqüência:
   $$s[n] = s[n] - \left\{\frac{d[n-1] + d[n]}{4}\right\}$$

2. Reconstrução

   - Computar:
   $$s[n] = s[n] + \left\{\frac{d[n-1] + d[n]}{4}\right\}$$
   $$d[n] = \left\{\frac{s[n] + s[n+1]}{2}\right\} - d[n]$$

   - Unir:
   $$x[2n] = s[n]$$
   $$x[2n+1] = d[n]$$

4.4.2.5 Transformada S+P

A transformada S+P os quais não resulta em uma forma direta de filtros biortogonais de suporte compacto, é um exemplo típico de métodos de correção. Uma generalização do método de correção para *wavelets* inteiras pode ser obtida em [7]. Como visto anteriormente, a decomposição inicia com a aplicação da transformada S e em seguida atualiza os componentes passa alta. Vários parâmetros podem sem usados na atualização dos coeficientes passa alta [5, 7, 47, 51]. Usaremos aqui $\alpha_{-1} = -\frac{1}{4}$, $\alpha_0 = -\frac{1}{8}$, $\alpha_1 = \frac{3}{8}$ e $\beta_1 = \frac{1}{4}$.

1. Decomposição:

   - Dividir:
   $$\begin{aligned} s[n] &= x[2n] \\ d[n] &= x[2n+1] \end{aligned}$$

   - Computar:
   $$d^{(1)}[n] = d[n] - s[n]$$
   $$s[n] = s[n] + \left\{\frac{d^{(1)}[n]}{2}\right\}$$

   - Atualizar os componentes de alta freqüência:
   $$d[n] = d^{(1)}[n] + \begin{Bmatrix} \alpha_{-1}(s[n-2] - s[n-1]) + \alpha_0(s[n-1] - s[n]) \\ +\alpha_1(s[n] - s[n+1]) - \beta_1 d^{(1)}[n+1] \end{Bmatrix}$$

2. Reconstrução

   - Computar:
   $$d^{(1)}[n] = d[n] - \begin{Bmatrix} \alpha_{-1}(s[n-2] - s[n-1]) + \alpha_0(s[n-1] - s[n]) \\ +\alpha_1(s[n] - s[n+1]) - \beta_1 d[n+1] \end{Bmatrix}$$
   $$s[n] = s[n] - \left\{\frac{d^{(1)}[n]}{2}\right\}$$
   $$d[n] = d^{(1)}[n] - s[n]$$

   - Unir:
   $$\begin{aligned} x[2n] &= s[n] \\ x[2n+1] &= d[n] \end{aligned}$$

Para um determinado filtro *wavelet*, podemos começar aplicando a transformada *wavelet* de inteiros (linear ou não linear), tais como a transformada de Haar ou *wavelet* "preguiçosa" (*lazy wavelet*), e então usar alguma fórmula de atualização apropriada para obter uma transformação *wavelet* de inteiros desejada.

Nas transformações anteriores não foi mostrado como efetuar a decomposição nas extremidades dos sinal. Apresentaremos a seguir, de forma resumida como isso pode ser feito. Assumiremos $x[n]$ o sinal a ser transformado e N o números de componentes deste sinal. Os valores N_l (número de coeficientes passa baixa) e M_l (número de coeficientes passa alta) são definidos como [6, 7]

$$N_l = \begin{cases} \dfrac{N}{2}, & se\ N\ for\ par \\ \dfrac{N+1}{2}, & se\ N\ for\ impar \end{cases} \quad \text{e} \quad M_l = N - N_l. \tag{4.27}$$

Adotando a extensão simétrica nas extremidades do sinal consideraremos que:

1. se os atuais filtros biortogonais têm tamanho par, estenderemos as extremidades do sinal da seguinte maneira:

$$\begin{cases} x[-n] = x[n-1], & para\ n = 1,2,\cdots \\ x[N+n] = x[N-n-1], & para\ n = 0,1,\cdots \end{cases}, \text{ e} \tag{4.28}$$

2. se os filtros têm tamanho impar, faremos a seguinte extensão:

$$\begin{cases} x[-n] = x[n], & para\ n = 1,2,\cdots \\ x[N+n] = x[N-n-2], & para\ n = 0,1,\cdots \end{cases} \tag{4.29}$$

Exemplos mostrando o processamento do sinal nas extremidades do sinal e a prova da propriedade de preservação de precisão podem ser obtidos em [7].

Nas transformadas acima, todos os cálculos podem ser feitos *in-place*, ou seja, sobrescrever os coeficientes do sinal pelo resultado da transformada sem a necessidade de alocação adicional de memória.

5. Experimentos Realizados

No capítulo 3, descrevemos a transformada *wavelet* na forma contínua e discreta e a sua relação com bancos de filtros. Mostramos também como a transformada *wavelet* pode ser utilizada com o objetivo de permitir maior compressão dos dados de uma imagem. Depois em seguida, no capítulo 4, mostramos que a transformada *wavelet* pode ser adaptada, através do mapeamento de inteiros para inteiros, para permitir a decomposição e a reconstrução da imagem sem que haja perda de informação neste processo. Vimos ainda, que para permitir que o número de bits seja mantido, sem a necessidade de bits adicionais para armazenar os coeficientes transformados, a transformada *wavelet* com a PPP pode ser empregada.

Neste capítulo, apresentaremos os resultados experimentais obtidos na compressão das projeções de tomografia computadorizada, baseados na transformada *wavelet* de inteiros para inteiros através da fatoração da transformada *wavelet* em passos *lifting*, conforme descritos no final do capítulo anterior (seção 4.4.2).

Inicialmente, descreveremos de forma resumida o processo de aquisição das projeções de tomografia computadorizada e o aspecto de uma das imagens de entrada que utilizamos no desenvolvimento deste trabalho (no apêndice 3 são apresentadas todas as imagens). Em seguida descreveremos o funcionamento da codificação aritmética que é utilizada após a aplicação da transformada *wavelet* com objetivo de reduzir o número de bits necessários para armazenar os dados. Realizamos também testes comparativos com a codificação de Huffman, porém o leitor interessando em mais detalhes da codificação de Huffman pode consultar [21, 25, 39]. Finalmente, no final deste capítulo apresentamos tabelas comparando os desempenhos obtidos pelos métodos de compressão utilizados.

5.1 Tomografia de Raio-X (CT)

A tomografia computadorizada (CT) é um método de radiografia que foi introduzida nos anos 60 para aplicações neurológicas, mas rapidamente com o avanço da tecnologia, permitiu examinar outras partes do corpo também. Allan McLeod Cormack investigou os assuntos numéricos envolvidos na tomografia computadorizada, e construiu o primeira máquina de CT ainda nos anos 60. Godfrey N. Hounsfield construiu o primeiro tomógrafo para imagens do cérebro humano. Hounsfield juntamente com Cormack receberam o Prêmio Nobel de Fisiologia e Medicina em 1979 [41]. A CT tem desde então estendido para aplicações industriais também.

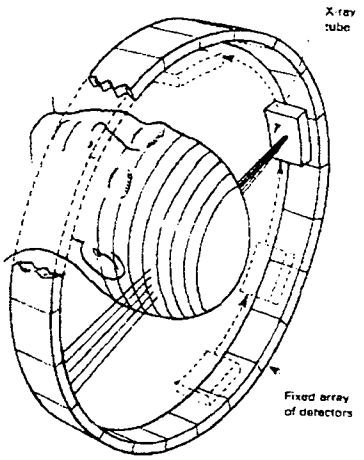


Figura 5.1 – Representação do funcionamento de um tomógrafo de raio-x.

O termo tomografia é usado para se referir a qualquer método de obtenção de imagem do interior de um determinado corpo a partir de medidas realizadas externamente. O termo "*comput(eriz)ed tomography*", ou somente CT, caiu no uso comum e até hoje se refere ao tomógrafo de raio-x. Existem várias outras técnicas de tomografia baseadas em outros princípios físicos, tais como: tomografia de ressonância magnética, *Magnetic Ressonance Imaging* (MRI), a tomografia de emissão de pósitrons, *Positron Emission Tomography* (PET), tomografia computadorizada de emissão simples de fótons, *Single Photon Emission*

Computed Tomography (SPECT), a tomografia com ultra-som, e a tomografia de impedância elétrica, *Electrical Impedance Tomography* (EIT).

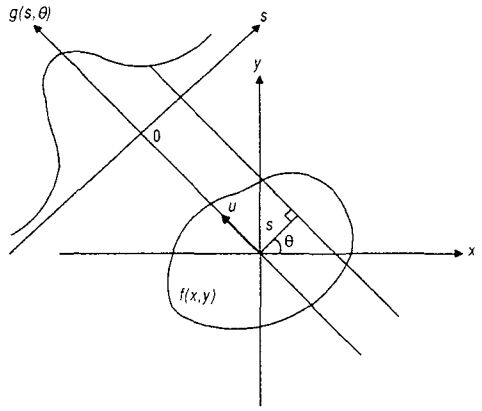


Figura 5.2 – Geometria da obtenção das projeções em CT.

A seguir será apresentado um modelo simplificado para CT de raio-x [28]. Seja $f(x,y)$ o coeficiente de absorção do objeto no ponto (x, y) numa faixa para algum valor fixo de z. Assumindo que os raios-x consistem de um feixe infinitamente fino e paralelo, a intensidade do feixe detectado do outro lado do objeto é dado por

$$I = I_0 \exp\left[-\int_L f(x,y)du\right] \qquad (5.1)$$

onde I_0 é a intensidade do feixe incidente, L é o caminho do raio e u é a distância ao longo de L (figura 5.2).

Considerando projeções em muitas direções diferentes, é possível reconstruir as características de densidade do corpo para cada ponto ou pequena região no interior do corpo.

A imagem obtida com a transformada de Radon de uma distribuição, que é a imagem que se obtém ao empilhar as linhas correspondentes as projeções obtidas para os vários ângulos, é muitas vezes referida como senograma [49].

A partir das projeções, obtidas com as medidas realizadas externamente ao corpo, deve-se reconstruir esses dados originais utilizando alguma técnica de reconstrução. Existem

vários algoritmos e técnicas que permitem a reconstrução da imagem a partir das projeções obtidas [26, 28, 41]. As técnicas que permitem a reconstrução da imagem também têm sido tópico de constante pesquisa.

O senograma, de uma tomografia computadorizada, pode ser visto como uma imagem digital constituída por níveis de cinza que variam de 0 (preto) até 65.535 (branco). A intensidade de cinza no senograma correspondem a atenuação sofrida pelo raio x entre a fonte e o detector. Os pontos com tons mais escuros correspondem às regiões que absorveram pouca porção do feixe de raio x e os mais claros indicam que ocorreu muita absorção, conforme as características do corpo que estava sendo analisado. Cada tecido do corpo humano têm diferentes coeficientes de atenuação (por exemplo, o pulmão tem um coeficiente de atenuação muito baixo, enquanto que os ossos têm um coeficientes de atenuação muito alto), dessa maneira os órgãos podem ser delineados e tecidos saudáveis podem ser distinguidos de tumores.

A figura 5.3 mostra, como exemplo, o senograma de uma CT, contendo 768 colunas e 90 linhas, antes de se fazer a sua reconstrução. As projeções foram obtidas utilizado um *phantom* de teste (objeto cilíndrico de acrílico contendo alguns furos no seu interior onde são adicionados diferentes elementos para análise). A imagem reconstruída a partir do senograma deste objeto pode ser observada na figura 5.4. Cada função seno no senograma resultará em um ponto na imagem reconstruída e cada senograma corresponde a uma fatia do corpo examinado.

Figura 5.3 – Imagem da CT antes da reconstrução.

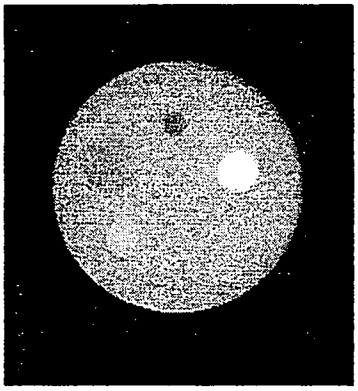


Figura 5.4 – Imagem da CT depois da reconstrução.

O tomógrafo que coletou as projeções da figura 5.3, consegue coletar 768 pontos em cada linha. O objeto foi rotacionado em 180 graus sendo coletadas informações de 2 em 2 graus. Com isso foi possível coletar as 90 linhas. Os feixes de raios x emitidos são captados por receptores que enviam os dados à um computador, que processa e transforma os dados, dando origem à imagem.

5.2 Codificação Aritmética

Os métodos de codificação de tamanho variável, ou codificação de entropia, mais utilizados na codificação dos coeficientes são: a codificação de Huffman [21, 25, 39] e a codificação aritmética [21, 36, 39, 70]. A codificação de Huffman é mais rápida enquanto que a codificação aritmética permite maior compressão. O primeiro é preferível quando os recursos de hardware são limitados, e o tempo de codificação/decodificação é o objetivo principal. A codificação aritmética é um pouco mais lenta do que a codificação de Huffman, mas é muito mais versátil e eficiente [51].

No código de Huffman o número de bits necessários para codificar uma informação têm que ser um número inteiro, e isso algumas vezes pode ser um problema. Por exemplo, se a probabilidade de ocorrência de um caracter é $\frac{1}{3}$, o melhor número de bits para codificar este caracter é de aproximadamente 1,58 bits. Com isso, o código de Huffman

precisará atribuir dois bits para o código, conduzindo para uma mensagem comprimida maior do que é teoricamente possível de acordo com o cálculo da entropia. Este problema aumenta quando a probabilidade de um caracter é muito alta.

Como o código de Huffman usa um número inteiro (k) de bits para cada símbolo, isto implica que nunca teremos k menor que 1. Numa imagem com duas cores (1 bit por *pixel*), como as usadas em fax, a compressão se torna impossível. A solução seria agrupar os bits em pacotes e então aplicar Huffman, porém isso impede o código de Huffman de ser um compressor universal.

A codificação aritmética substitui os símbolos de entrada com um código específico. Um fluxo de bits de símbolos de entrada é codificado em um único número de ponto flutuante maior ou igual a zero e menor do que 1. Este número pode ser unicamente decodificado para extrair os símbolos originais.

Há dois pontos fundamentais na codificação aritmética: a probabilidade de um símbolo e o limite de seu intervalo na codificação. As probabilidades dos símbolos de origem determinam a eficiência da compressão. Eles também determinam o limite do intervalo dos símbolos de origem para o processo de codificação. Estes limites de intervalo estão contidos dentro de um intervalo de 0 a 1. O limite do intervalo para o processo de codificação determina a saída comprimida.

O processo de codificação de um codificador aritmético pode ser melhor explicado através do exemplo abaixo:

Supondo que os símbolos de entrada são $\{a, b, c, d\}$ e a probabilidade destes símbolos são $\{0.1, 0.4, 0.2, 0.3\}$ respectivamente. Então baseado em suas probabilidades, o intervalo $[0, 1)$ pode ser dividido como 4 sub-intervalos: $[0, 0.1)$, $[0.1, 0.5)$, $[0.5, 0.7)$ e $[0.7, 1)$, onde $[x, y)$ denota um intervalo fechado a esquerda e aberto a direita, os quais inclui x mas

exclui *y*. A informação acima pode ser resumida na tabela abaixo:

Símbolos	*a*	*b*	*c*	*d*
Probabilidades	0.1	0.4	0.2	0.3
Intervalo Inicial	[0, 0.1)	[0.1, 0.5)	[0.5, 0.7)	[0.7, 1)

Tabela 5.1 – Representação dos símbolos e seus respectivos intervalos inciais.

Imagine transmitindo a mensagem "*cadacdb*". Inicialmente, tanto o codificador quanto o decodificador conhecem que o intervalo é [0, 1). Após ler o primeiro símbolo, *c*, da mensagem sabemos que o seu intervalo de codificação é [0.5, 0.7). Após este símbolo ter sido processado, o tamanho do modelo é modificado para o intervalo [0.5, 0.7), como mostrado na figura 5.5. O intervalo agora passa a ser [0.5, 0.7). Dividindo este novo intervalo, de acordo com as probabilidades de ocorrências dos símbolos, teremos os seguintes sub-intervalos: [0.5, 0.52), [0.52, 0.6), [0.6, 0.64), [0.64, 0.7).

O segundo símbolo da mensagem é o *a*. Este símbolo será codificado tomando o primeiro décimo do intervalo [0.5, 0.7). Portanto, teremos um novo intervalo que será [0.5, 0.52). Similarmente, codificando o terceiro símbolo, *d*, nós temos um novo intervalo [0.514, 0.52). Depois codificando o quarto símbolo, *a*, o novo intervalo é [0.514, 0.5146). E assim por diante até que após a codificação do último símbolo, *b*, teremos o intervalo [0.5143876, 0.514402). A saída desta mensagem pode ser qualquer número que esteja dentro deste último intervalo.

Novo caracter	Limite inferior	Limite superior
	0.0	1.0
c	0.5	0.7
a	0.5	0.52
d	0.514	0.52
a	0.514	0.5146
c	0.5143	0.51442
d	0.514384	0.51442
b	0.5143876	0.514402

Tabela 5.2 – Processo de codificação do exemplo.

Visualmente, podemos representar o processo acima através da figura abaixo:

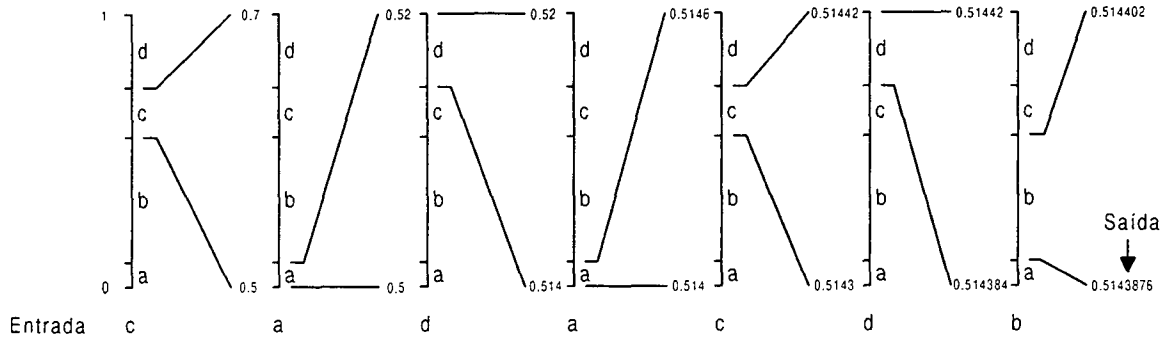


Figura 5.5 – Representação do processo de codificação aritmética.

Dado este esquema de codificação, é relativamente fácil ver como o processo de decodificação funciona. Inicialmente, o decodificador verifica em qual intervalo o primeiro símbolo da mensagem se encontra. Desde que 0.5143876 está dentro do intervalo [0.5, 0.7) o primeiro caracter da mensagem deverá ser o *c*. Como conhecemos o limite inferior e superior de *c* que é [0.5, 0.7), revertemos o processo que gerou ele. Primeiro, subtraímos pelo limite inferior de *c* que implicará em 0.0143876. Então fazemos a divisão pelo tamanho da largura do intervalo de *c*, ou seja, 0.2. Isto resulta em 0,071938. Podemos verificar que o próximo símbolo será o *a*, pois 0,071938 se encontra no intervalo [0, 0.1). Em seguida teremos o valor 0,71938, que se encontra no intervalo [0.7, 1) resultando no símbolo *d*. E assim por diante, todos os símbolos da mensagem inicial serão recuperados.

Número Codificado	Símbolo de Saída	Menor	Maior	Tam. Intervalo
0.5143876	*c*	0.5	0.7	0.2
0,071938	*a*	0.0	0.1	0.1
0,71938	*d*	0.7	1.0	0.3
0,0646	*a*	0.0	0.1	0.1
0,646	*c*	0.5	0.7	0.2
0,73	*d*	0.7	1.0	0.3
0,1	*b*	0.1	0.5	0.4
0.0				

Tabela 5.3 – Processo de decodificação do exemplo.

A ordem do modelo, refere-se ao número de símbolos anteriores que são usados para calcular a probabilidade de cada símbolo de entrada. Neste exemplo, usamos o modelo de ordem-0 que calcula a probabilidade de cada símbolo independentemente de qualquer símbolo anterior.

5.3 Esquema de Compressão

Nesta seção, descrevemos em alguns detalhes o processo de compressão utilizado na realização deste trabalho. O processo é ilustrado na figura 5.6. Nesta figura, foi acrescentado duas novas etapas que não estão sendo consideradas neste trabalho, que teriam a função de reconstruir as imagens de tomografia a partir dos senogramas, permitindo assim a sua análise.

As informações fornecidas pelo tomógrafo ao computador constitui de valores inteiros no formato binário. Estes valores variam de 0 a 65535, ou seja, 16 bits não sinalizados. Dessa maneira, cada senograma é armazenado no computador usando 138.240 bytes.

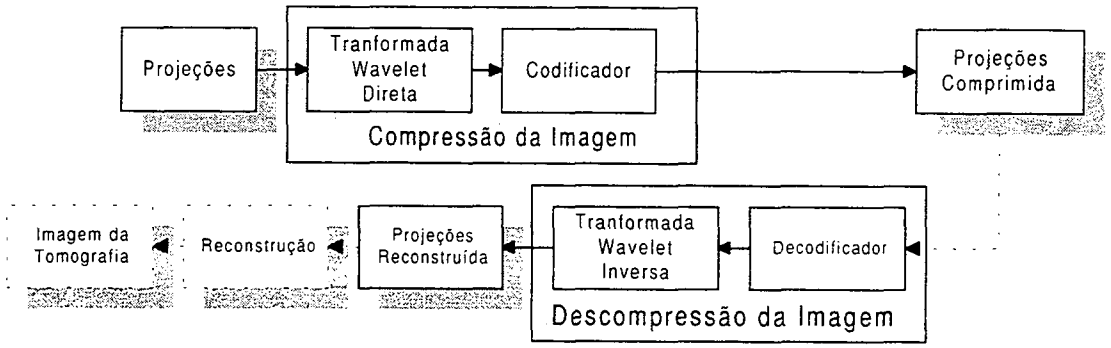


Figura 5.6 – Esquema de compressão usado no experimentos.

Neste trabalho estamos considerando apenas as transformadas *wavelet* de inteiros apresentadas no final do capítulo anterior (seção 4.4.2), isto é, a transformada S ou (1, 1), S+P, (1, 3), (2, 6) e (5, 3), os quais possuem a propriedade de preservação de precisão (PPP).

Implementamos as transformadas *wavelet* bidimensionais usando a linguagem C (Borland C++ versão 3.1), o qual permite portabilidade para outros sistemas operacionais com pequenas ou nenhuma modificação. Estamos usando a decomposição *wavelet* padrão e o armazenamento dos coeficientes da transformada *wavelet* sobrescrevem os dados originais na mesma localização de memória (*in-place*), não necessitando de memória auxiliar como ocorre com a ordem clássica de Mallat. O código permite a entrada de diferentes tamanhos de imagens, apesar de estarmos usando apenas imagens de 768×90. O usuário pode optar também pelo número de níveis de decomposição desejado.

No presente trabalho, estamos decompondo as imagens em 5 níveis de resolução, tanto nas linhas quanto nas colunas. Em cada nível de decomposição o número de coeficientes são reduzidos pela metade, pelo fato de que apenas os coeficientes passa baixa serão usados na aplicação da transformada no próximo nível de resolução. Como as imagens possuem 90 linhas, 5 níveis de decomposição são suficientes para reduzir para apenas 3 *pixels*, quando a transformada bidimensional é aplicada nas colunas da imagem.

Na fase de codificação, usamos a codificação aritmética [21, 36, 39, 70]. Apenas para fins de comparação, usamos também a codificação de Huffman [21, 25, 39].

5.4 Resultados

Mostraremos agora os resultados obtidos com o esquema de compressão apresentado na seção anterior. Neste trabalho estamos considerando dez imagens para testes. As imagens e a reconstrução delas estão listadas no Apêndice 3. Os nomes originais das imagens foram mantidos.

A tabela 5.4 ilustra a medida de entropia das imagens das projeções de tomografia. H_0 é a entropia de primeira ordem, onde a probabilidade de ocorrência de cada símbolo é calculado independentemente de qualquer símbolo anterior.

Imagem	H_0
am4	6,278606
amos_10	6,943776
babacu	7,030203
babacu2	7,033890
cabo2_3	6,186742
concret2	6,340983
concreto	6,681465
dente01	7,118583
dente02	7,111283
dente03	7,129697

Tabela 5.4 – Medidas de entropia de primeira ordem.

A tabela abaixo corresponde ao cálculo da entropia de primeira ordem dos coeficientes das diferentes transformadas *wavelet* de inteiros em bits por *pixel* (bpp). A transformada foi aplicada levando em consideração cada informação na sua forma original, ou seja, com 16 bits. Após a aplicação da transformada o número de bits se manteve o mesmo em virtude da PPP, discutido no capítulo anterior.

Imagem	S (1, 1)	(1, 3)	(2, 6)	(5, 3)	S+P
am4	5,3743	5,8067	5,5647	5,9526	5,2762
amos_10	5,6316	6,1294	5,7812	6,2938	5,5913
babacu	5,7879	6,1451	6,0057	6,3227	5,6493
babacu2	5,7470	6,0767	5,9714	6,2795	5,6160
cabo2_3	5,1417	5,3367	5,3053	5,3663	5,1329
concret2	5,6945	5,6468	5,9319	5,8494	5,6692
concreto	5,4417	5,6623	5,6191	5,7211	5,4511
dente01	5,9276	6,3643	6,1776	6,6926	5,8256
dente02	5,7745	6,2726	5,9607	6,6254	5,6787
dente03	5,7821	6,4135	5,9779	6,7798	5,6906
média	5,6303	5,9854	5,8296	6,1883	5,5581

Tabela 5.5 – Comparação de diferentes transformadas *wavelet* (em bpp)

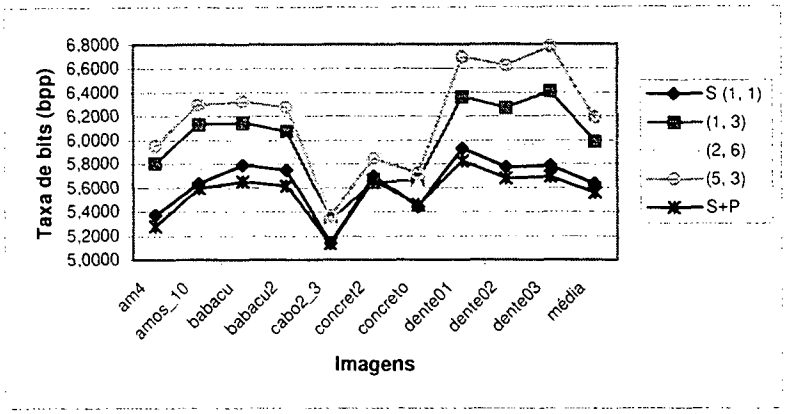


Figura 5.7 – Gráfico com a taxa de bits por *pixel*.

Em seguida, temos uma tabela comparativa usando a transformada S+P com os codificadores aritmético e de Huffman. Escolhemos a transformada S+P pelo fato de ter apresentado melhores resultados em todas as imagens testadas, em termos do cálculo da entropia dos coeficientes transformados. Apresentamos também, uma comparação com GZIP[2]. O GZIP (Gnu ZIP) é um software desenvolvido sobre a licença GNU e distribuído gratuitamente para compressão de dados de propósito geral. O GZIP atualmente usa o algoritmo Lempel-Ziv (LZ77) [73].

Imagem	**S+P e Cod. Aritmética**	**S+P e Cod. Huffman**	**GZIP**
am4	91.746	92.124	110.515
amos_10	97.432	97.676	126.260
babacu	98.273	98.468	125.625
babacu2	97.690	97.879	125.579
cabo2_3	89.366	89.583	108.443
concret2	98.493	98.841	111.614
concreto	94.780	95.019	121.512
dente01	101.376	101.750	126.261
dente02	98.764	98.991	126.059
dente03	98.994	99.207	126.331
média	96.691	96.954	120.820

Tabela 5.6 – Tamanho dos arquivos comprimidos (em bytes).

[2] ftp://prep.ai.mit.edu/pub/gnu/gzip

Podemos observar que a média de compressão com a transformada *wavelet* é bem superior ao GZIP. Outros programas compressores de dados de propósito geral (baseados no algoritmo LZ) apresentam taxas de compressão muito próximas do GZIP.

Com objetivo de explorar mais a correlação dos dados das projeções, aplicamos a transformada nos bytes mais significativos e nos bytes menos significativos dos dados de entrada. Os dados da tabela abaixo são relativos a média do cálculo da entropia aplicado nas duas imagens no qual o senograma foi dividido.

Imagem	**S (1, 1)**	**(1, 3)**	**(2, 6)**	**(5, 3)**	**S+P**
am4	4,7993	4,9215	4,9619	5,1247	5,1604
amos_10	4,8842	5,0028	5,1399	5,3542	5,3173
babacu	5,1022	5,1432	5,3044	5,4450	5,5231
babacu2	5,0671	5,0937	5,2953	5,4654	5,5079
cabo2_3	4,4755	4,5603	4,5595	4,7518	4,6740
concret2	5,0261	4,8820	5,1323	5,2249	5,2617
concreto	4,7749	4,8496	4,8920	5,0867	5,0128
dente01	5,3005	5,3185	5,5226	5,7201	5,8055
dente02	5,0858	5,1802	5,3492	5,5841	5,6498
dente03	5,0964	5,1855	5,3280	5,5550	5,5873
média	4,9612	5,0137	5,1485	5,3312	5,3500

Tabela 5.7 – Comparação de diferentes transformadas *wavelet* de inteiros (bits/*pixel*).

O custo computacional neste caso é maior, porém tem-se um ganho maior em termos de taxa de compressão. Nos bytes menos significativos, o ganho é muito pouco quando comparado com a entropia dos dados. Todavia, os bytes mais significativos fornecem uma taxa de compressão muito alta. Mesmo aplicando a transformada apenas na imagem correspondente aos bytes mais significativos, consegue-se aumentar a taxa de compressão em relação a aplicação da transformada com dados de 16 bits.

Nesse caso, a transformada que apresentou melhores resultados foi a transformada S. De um modo geral, os filtros com menos coeficientes apresentaram melhores resultados. Uma representação gráfica da tabela anterior é apresentada a seguir.

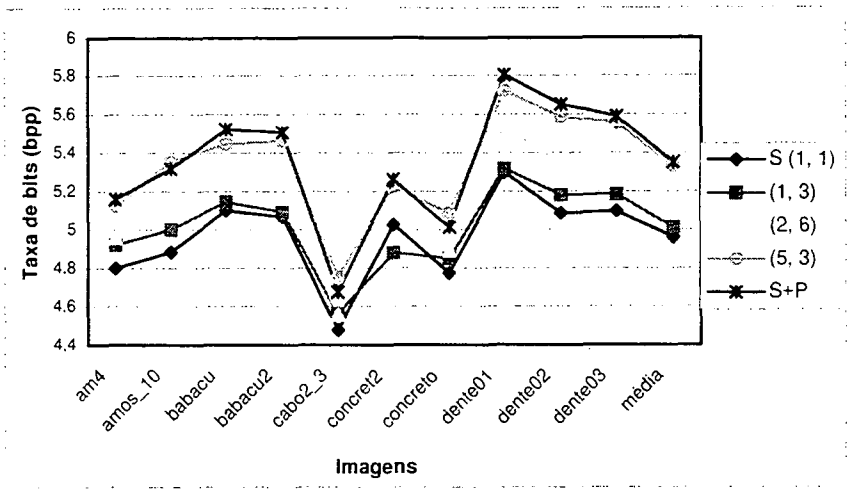


Figura 5.8 – Gráfico com a taxa de bits das imagens.

A próxima tabela apresenta uma avaliação em termos percentuais da taxa de compressão. A fórmula usada no cálculo foi a mesma já apresentada no capítulo 2.

Imagem	**S (1, 1)**	**(1, 3)**	**(2, 6)**	**(5, 3)**	**S+P**
am4	40,0%	38,5%	38,0%	35,9%	35,5%
amos_10	38,9%	37,5%	35,8%	33,1%	33,5%
babacu	36,2%	35,7%	33,7%	31,9%	31,0%
babacu2	36,7%	36,3%	33,8%	31,7%	31,2%
cabo2_3	44,1%	43,0%	43,0%	40,6%	41,6%
concret2	37,2%	39,0%	35,8%	34,7%	34,2%
concreto	40,3%	39,4%	38,8%	36,4%	37,3%
dente01	33,7%	33,5%	31,0%	28,5%	27,4%
dente02	36,4%	35,2%	33,1%	30,2%	29,4%
dente03	36,3%	35,2%	33,4%	30,6%	30,2%
média	38,0%	37,3%	35,6%	33,4%	33,1%

Tabela 5.8 – Taxas de compressão.

Para uma avaliação mais detalhada do tamanho dos dados comprimidos, apresentamos a tabela abaixo. Nela temos os melhores resultados obtidos com a transformada *wavelet* de inteiros nos dois casos apresentados anteriormente (S+P computando com 16 bits e S computando com 8 bits) juntamente com os resultados obtidos com o LJPG[3] (*lossless* JPG) [13, 14, 67] e o GZIP. A coluna da entropia na tabela, corresponde ao tamanho do

[3] Huang, K., Smith, B., "Lossless JPEG Codec Version 1.0", June, 1994.
http://www.cs.cornell.edu/Info/Projects/zeno/Projects/LJPG.html

arquivo comprimido que pode ser obtido levando em consideração apenas a redundância de código baseado no cálculo da entropia de primeira ordem. Apenas a aplicação da codificação aritmética, codificação de Huffman ou codificação de Shannon-Fano apresentam valores próximos desses.

Imagem	S e Cod. Aritmética	S+P e Cod. Aritmética.	LJPG	GZIP	Entropia
am4	86.529	91.746	86.150	110.515	116.275
amos_10	87.321	97.432	91.515	126.260	127.494
babacu	91.398	98.273	91.997	125.625	127.175
babacu2	90.566	97.690	91.779	125.579	127.189
cabo2_3	81.154	89.366	85.319	108.443	115.185
concret2	91.373	98.493	87.556	111.614	118.624
concreto	86.450	94.780	87.671	121.512	123.971
dente01	95.104	101.376	93.972	126.261	127.745
dente02	92.119	98.764	92.386	126.059	127.676
dente03	92.114	98.994	92.375	126.331	127.927
média	89.413	96.691	90.072	120.820	123.926

Tabela 5.9 – Comparação em tamanho (bytes) com diferentes métodos.

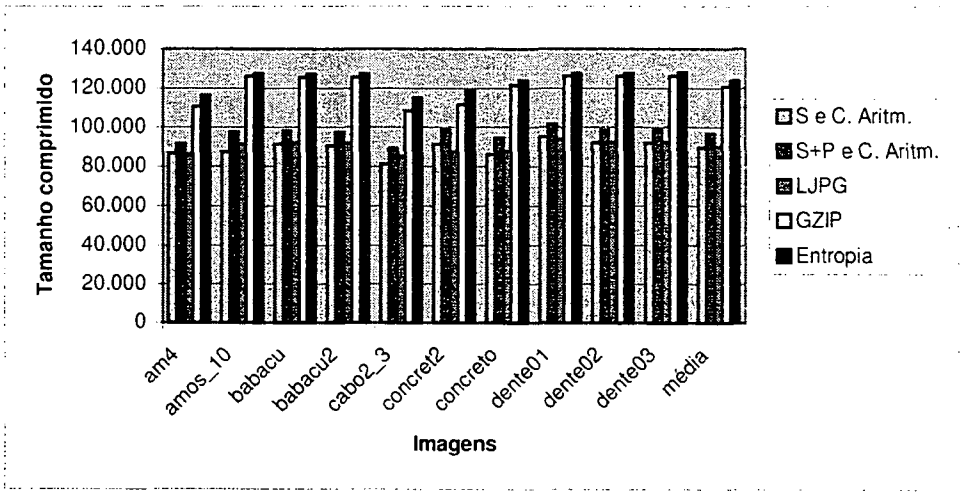


Figura 5.9 – Gráfico comparativo dos resultados obtidos.

5.5 Considerações finais

Realizamos também alguns testes com as transformadas *wavelet* de inteiros para inteiros usadas em [4, 5] (ver apêndice 2.7). Todavia, em virtude das características

apresentadas nos senogramas, após a aplicação da transformada muitos coeficientes apresentavam valores muito altos ou baixos e dessa maneira havia a necessidade de aumentar o número de bits para armazenar estes coeficientes, com isso diminuía a performance dos codificadores de entropia. O mesmo ocorreu com LJPG aplicado ao senograma dividido em duas imagens. Na imagem correspondente ao byte mais significativo ocorre uma taxa de compressão muito alta, porém na imagem correspondente ao byte menos significativo apresenta uma taxa de compressão muito pequena ou uma taxa de compressão negativa. O ideal nesses casos seria aplicar a transformada *wavelet* apenas em uma da imagens e codificar a outra apenas com codificadores de entropia.

Outro teste realizado foi o uso da codificação RLE antes de aplicar a codificação aritmética, porém o desempenho não foi satisfatório pelo fato de estarmos considerando apenas compressão sem perdas.

Vale a pena ressaltar também que não estamos levando em consideração o desempenho da transformada *wavelet* em relação a tempo de processamento. A literatura demostra que a transformada *wavelet* é muito eficiente computacionalmente. Procura-se usar os coeficientes racionais para os filtros, principalmente coeficientes racionais diádicos. Multiplicações com valores de potência de 2 correspondem a deslocamento de bits que é uma operação muito rápida. O fato de usar aritmética de inteiros também contribui para diminuir o tempo de processamento.

6. Conclusão

As imagens digitais geram um grande volume de dados, e para propósitos de transmissão ou armazenamento é desejável comprimir estes dados. Certos tipos de dados não podem sofrer perdas ocasionadas pela compressão/descompressão, como é o caso da projeção de tomografia (senograma). Perdas de informações nestas imagens podem resultar na inserção de artefatos, e no caso de imagens médicas pode prejudicar o diagnóstico.

Neste trabalho avaliamos uma das inúmeras aplicação da transformada *wavelet* em processamento de imagens, mais especificamente na compressão de projeções de tomografia computadorizada. Uma vantagem de comprimir as projeções de CT em vez das imagens já reconstruídas é a preservação dos dados originais. Uma vez que os dados descomprimidos são iguais aos originais obtidos pelo tomógrafo, é possível fazer, quando necessária, a reconstrução da imagem utilizando qualquer técnica de reconstrução disponível.

Os resultados obtidos neste trabalho mostraram que com a transformada *wavelet* é possível obter uma redução considerável no tamanho dos arquivos. Na média, a taxa de compressão das projeções ficou acima de 35%. Quando comparado com o algoritmo de propósito geral (GZIP), os resultados apontam para um ganho de 22,7%. Em relação ao LJPG, o ganho apresentado foi mínimo, porém existem vários aprimoramentos que podem ser realizados com objetivo de melhorar os resultados obtidos com a transformada *wavelet*. Mais adiante, citaremos algumas técnicas que podem ser aplicadas neste sentido.

Quando considerando os dados na sua forma original, ou seja, dados com 16 bits a transformada S+P apresentou melhores resultados para todas as imagens, mas dividindo o senograma em duas imagens os melhores resultados foram obtidos na sua maioria com a transformada S. Filtros com menos coeficientes apresentaram melhores resultados, pois as

imagens apresentam uma grande variação no valor de seus coeficientes na região correspondente a amostra que está sendo analisada.

Como mostrado em [4, 5], filtros *wavelet* com mais momentos de anulação na análise geralmente apresentam melhores resultados com imagens naturais e suaves e não tão bom com imagens com muitas bordas e componentes de alta freqüência. Por outro lado, filtros com baixa ordem como a transformada S geralmente apresentam os piores resultados, especialmente com imagens naturais. Todavia, realizam melhor compressão em imagens com muitas bordas e componentes de alta freqüência.

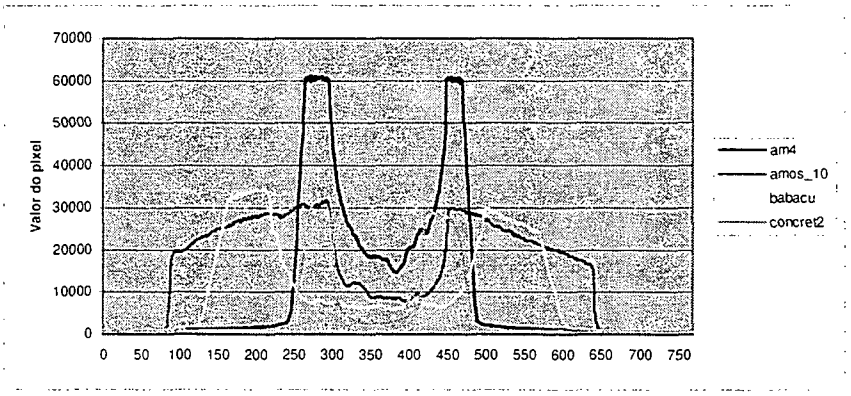


Figura 6.1 – Representação de linhas das imagens.

O gráfico da figura 6.1 mostra, como exemplo, a primeira linha de quatro senogramas usados nos teste. As demais linhas dos senogramas apresentam características semelhantes. Nelas é possível observar que existem três regiões que podem ser analisadas de forma diferente. Os primeiros coeficientes e os últimos apresentam uma pequena variação no seu valor. Dessa maneira, filtros com mais momentos de anulação tendem a apresentar melhores resultados nessas duas regiões. Na região intermediária ocorre uma variação muito grande no valor dos coeficientes, com isso, os filtros com menos momentos de anulação apresentam um ganho maior na codificação.

A transformada *wavelet* mostra-se como uma poderosa ferramenta para compressão de imagens. Entre as vantagens da transformada *wavelet* podemos citar a decomposição em multiresolução, localidade no domínio tempo/espaço e no domínio da freqüência, funções bases suaves (controladas pelos momentos de anulação), boa aproximação em $L^2(\mathbf{R})$, algoritmos com baixa complexidade computacional através da FWT ($O(n)$), compressão com perdas ou sem perdas dependendo da aplicação, realização de altas taxas de compressão mantendo a qualidade da imagem e transmissão progressiva.

Como sugestões de propostas para trabalhos futuros a serem consideradas, visando melhorar ainda mais o desempenho em termos de compressão das projeções de tomografia computadorizada podemos citar:

- Utilização de alguma técnica de modelagem nos coeficientes transformados para melhorar o desempenho dos codificadores de entropia. Técnicas tais como: aplicação da codificação *zerotree* (EZW) [55] na codificação dos coeficientes *wavelets*, DPCM ou algoritmos que explorem a redundância interpixel desses coeficientes. Podemos considerar também outras trajetórias a serem percorridas na codificação, tais como a curva de Hilbert [40, 58] ou codificar cada quadrante da imagem transformada separadamente [4, 5, 51]. Essas técnicas tendem a aumentar o tempo de compressão/descompressão, porém deve apresentar resultados melhores do que os obtidos aqui;
- Avaliação do ganho obtido com a compressão das projeções em relação as imagens já reconstruídas e com outros métodos de compressão de imagens sem perdas existentes;
- Otimização do código da transformada *wavelet* direta e inversa, objetivando maior rapidez computacional na sua aplicação.

- Aplicação de outros filtros *wavelet* que se adaptem melhor às características presentes nas projeções, usando o *lifting scheme* ou o método de correção; e
- Desenvolvimento de um ambiente integrado que permita realizar a compressão/descompressão das projeções e também efetuar a reconstrução da imagem para análise.

Apêndice 1 - O Espaço $L^2(R)$

Os espaços vetoriais $L^P(\mathbf{R})$, considerando aqui apenas o caso em que p = 1, 2, são definidos por [17, 19, 38]:

$$L^P(\mathbf{R}) = \{ f: \mathbf{R} \rightarrow \mathbf{R}; f \text{ é mensurável e } |f|^P \text{ é integrável sobre } \mathbf{R} \}. \quad (1.2)$$

O espaço $L^1(\mathbf{R})$ é um espaço Banach, munido da norma:

$$\|f\|^1 = \int_{-\infty}^{\infty} |f(t)|\, dt \quad (1.3)$$

O espaço $L^2(\mathbf{R})$ representa o espaço de funções quadraticamente integráveis, isto é, de energia finita. A energia da função f é finita ao longo de todo eixo dos reais

$$\int_{-\infty}^{\infty} |f(t)|^2 dt < \infty. \quad (1.4)$$

O produto interno de duas funções $f(t) \in L^2(\mathbf{R})$ e $g(t) \in L^2(\mathbf{R})$ é definido por

$$\langle f, g \rangle = \int_{-\infty}^{\infty} f(t).g^*(t) dt, \quad (1.5)$$

e f e g são ditas serem ortogonais quando $\langle f, g \rangle = 0$.

A transformada de Fourier de uma função $f(t) \in L^2(\mathbf{R})$ é definida como

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt. \quad (1.6)$$

A norma de $f(t)$ em $L^2(\mathbf{R})$ é definida por

$$\|f\|^2 = \int_{-\infty}^{\infty} |f(t)|^2 dt \quad (1.7)$$

$$\|f\| = \sqrt{\langle f, f \rangle}. \quad (1.8)$$

A convolução de duas funções $f(t)$ e $g(t) \in L^2(\mathbf{R})$ é definido por

$$f(t) * g(t) = \int_{-\infty}^{\infty} f(u) g(t-u) du. \quad (1.9)$$

Apêndice 2 - *Wavelets* Biortogonais de Cohen-Daubechies-Feauveau

A família de *wavelets* biortogonais de Cohen-Daubechies-Feaveau (CDF) [9] têm algumas propriedades interessantes, tais como:

- A função de escala $\varphi(t)$ é sempre simétrica e a função *wavelet* $\psi(t)$ é sempre simétrica ou antisimétrica.
- Os filtros *wavelet* têm suporte finito.
- Os coeficientes dos filtros *wavelet* são da forma $\frac{z}{2^n}$, para $z \in \mathbf{Z}$ e $n \in \mathbf{N}$. Dessa maneira todas as divisões podem ser implementadas usando deslocamentos binários, que torna mais rápido a computação. Infelizmente os coeficientes dos passos *lifting* não são sempre desta forma.

Os filtros de análise para algumas das *wavelets* biortogonais de Cohen-daubechies-Feauveau (CDF) [9, 65] são mostrados abaixo para $x \in \{1, 3, 5\}$ ou $x \in \{2, 4, 6\}$. A notação $(N, \tilde{N})$ representa a transformada com N e $\tilde{N}$ momentos de anulação nos filtros passa alta de análise e síntese respectivamente.

2.1 CDF (1, x)

$$\tilde{g}_{(1,x)} : \frac{\sqrt{2}}{2} \cdot (1,-1) \qquad (2.1)$$

$$\tilde{h}_{(1,1)} : \frac{\sqrt{2}}{2} \cdot (1,1) \qquad (2.2)$$

$$\tilde{h}_{(1,3)} : \frac{\sqrt{2}}{16} \cdot (-1,1,8,8,1,-1) \qquad (2.3)$$

$$\tilde{h}_{(1,5)} : \frac{\sqrt{2}}{256} \cdot (3,-3,-22,22,128,128,22,-22,-3,3) \qquad (2.4)$$

Os passos *lifting* são:

$$d[n] \leftarrow d[n] - s[n] \tag{2.5}$$

$$s[n] \xleftarrow{(1,1)} s[n] + \frac{1}{2} d[n] \tag{2.6}$$

$$s[n] \xleftarrow{(1,3)} s[n] - \frac{1}{16}(-d[n-1] - 8d[n] + d[n+1]) \tag{2.7}$$

$$s[n] \xleftarrow{(1,5)} s[n] - \frac{1}{256}(3d[n-2] - 22d[n-1] - 128d[n] + 22d[n+1] - 3d[n+2]) \tag{2.8}$$

Os fatores de normalização são:

$$n_L = \sqrt{2} \tag{2.9}$$

$$n_H = \frac{\sqrt{2}}{2} \tag{2.10}$$

2.2 CDF (2, x)

$$\tilde{g}_{(2,x)} : \frac{\sqrt{2}}{4} \cdot (1, -2, 1) \tag{2.11}$$

$$\tilde{h}_{(2,2)} : \frac{\sqrt{2}}{8} \cdot (-1, 2, 6, 2, -1) \tag{2.12}$$

$$\tilde{h}_{(2,4)} : \frac{\sqrt{2}}{128} \cdot (3, -6, -16, 38, 90, 38, -16, -6, 3) \tag{2.13}$$

$$\tilde{h}_{(2,6)} : \frac{\sqrt{2}}{1024} \cdot (-5, 10, 34, -78, -123, 324, 700, 324, -123, -78, 34, 10, -5) \tag{2.14}$$

Os passos *lifting* são:

$$d[n] \leftarrow d[n] - \frac{1}{2}(s[n] + s[n+1]) \tag{2.15}$$

$$s[n] \xleftarrow{(2,2)} s[n] - \frac{1}{4}(-d[n-1] - d[n]) \tag{2.16}$$

$$s[n] \xleftarrow{(2,4)} s[n] - \frac{1}{64}(3d[n-2] - 19d[n-1] - 19d[n] + 3d[n+1]) \tag{2.17}$$

$$s[n] \xleftarrow{(2,6)} s[n] - \frac{1}{512}\begin{pmatrix} -5d[n-3]+39d[n-2]-162d[n-1] \\ -162d[n]+39d[n+1]-5d[n+2] \end{pmatrix} \quad (2.18)$$

Os fatores de normalização são:

$$n_L = \sqrt{2} \quad (2.19)$$

$$n_H = \frac{\sqrt{2}}{2} \quad (2.20)$$

2.3 CDF (3, x)

$$\tilde{g}_{(3,x)} : \frac{\sqrt{2}}{8} \cdot (-1,3,-3,1) \quad (2.21)$$

$$\tilde{h}_{(3,1)} : \frac{\sqrt{2}}{4} \cdot (-1,3,3,-1) \quad (2.22)$$

$$\tilde{h}_{(3,3)} : \frac{\sqrt{2}}{64} \cdot (3,-9,-7,45,45,-7,-9,3) \quad (2.23)$$

$$\tilde{h}_{(3,5)} : \frac{\sqrt{2}}{512} \cdot (-5,15,19,-97,-26,350,-26,-97,19,15,-5) \quad (2.24)$$

Os passos *lifting* são:

$$s[n] \leftarrow s[n] - \frac{1}{3}d[n-1] \quad (2.25)$$

$$d[n] \leftarrow d[n] - \frac{1}{8}(9s[n]+3s[n+1]) \quad (2.26)$$

$$s[n] \xleftarrow{(3,1)} s[n] + \frac{4}{9}d[n] \quad (2.27)$$

$$s[n] \xleftarrow{(3,3)} s[n] - \frac{1}{36}(-3d[n-1]-16d[n]+3d[n+1]) \quad (2.28)$$

$$s[n] \xleftarrow{(3,5)} s[n] - \frac{1}{288}\begin{pmatrix} 5d[n-2]-34d[n-1]-128d[n] \\ +34d[n+1]-5d[n+2] \end{pmatrix} \quad (2.29)$$

Os fatores de normalização são:

$$n_L = \frac{3\sqrt{2}}{2} \tag{2.30}$$

$$n_H = \frac{\sqrt{2}}{3} \tag{2.31}$$

2.4 CDF (4, x)

$$\tilde{g}_{(4,x)} : \frac{\sqrt{2}}{16} \cdot (-1,4,-6,4,-1) \tag{2.32}$$

$$\tilde{h}_{(4,2)} : \frac{\sqrt{2}}{32} \cdot (3,-12,5,40,5,-12,3) \tag{2.33}$$

$$\tilde{h}_{(4,4)} : \frac{\sqrt{2}}{512} \cdot (-10,40,-2,-192,140,560,140,-192,-2,40,-10) \tag{2.34}$$

$$\tilde{h}_{(4,6)} : \frac{\sqrt{2}}{8192} \cdot \begin{pmatrix} 35,-140,-55,920,-557,-2932,2625,8400, \\ 2625,-2932,-557,920,-55,-140,35 \end{pmatrix} \tag{2.35}$$

Os passos *lifting* são:

$$s[n] \leftarrow s[n] - \frac{1}{4}(d[n-1] + d[n]) \tag{2.36}$$

$$d[n] \leftarrow d[n] - (s[n] + s[n+1]) \tag{2.37}$$

$$s[n] \xleftarrow{(4,2)} s[n] - \frac{1}{16}(-3d[n-1] - 3d[n]) \tag{2.38}$$

$$s[n] \xleftarrow{(4,4)} s[n] - \frac{1}{128}(5d[n-2] - 29d[n-1] - 29d[n] + 5d[n+1]) \tag{2.39}$$

$$s_i \xleftarrow{(4,6)} s_i - \frac{1}{4096}(-35d_{i-3} + 265d_{i-2} - 998d_{i-1} - 998d_i + 265d_{i+1} - 35d_{i+2}) \tag{2.40}$$

Os fatores de normalização são:

$$n_L = 2\sqrt{2} \tag{2.41}$$

$$n_H = \frac{\sqrt{2}}{4} \tag{2.42}$$

2.5 CDF (5, x)

$$\tilde{g}_{(5,x)} : \frac{\sqrt{2}}{32} \cdot (1,-5,10,-10,5,-1) \tag{2.43}$$

$$\tilde{h}_{(5,1)} : \frac{\sqrt{2}}{16} \cdot (3,-15,20,20,-15,3) \tag{2.44}$$

$$\tilde{h}_{(5,3)} : \frac{\sqrt{2}}{128} \cdot (-5,25,-26,-70,140,140,-70,-26,25,-5) \tag{2.45}$$

$$\tilde{h}_{(5,5)} : \frac{\sqrt{2}}{4096} \cdot \begin{pmatrix} 35,-175,120,800,-1357,-1575,4200, \\ 4200,-1575,-1357,800,120,-175,35 \end{pmatrix} \tag{2.46}$$

Os passos *lifting* são:

$$d[n] \leftarrow d[n] - \frac{1}{5} s[n] \tag{2.47}$$

$$s[n] \leftarrow s[n] - \frac{1}{24}(15d[n-1] + 5d[n]) \tag{2.48}$$

$$d[n] \leftarrow d[n] - \frac{1}{10}(15s[n] + 9s[n+1]) \tag{2.49}$$

$$s[n] \xleftarrow{(5,1)} s[n] + \frac{1}{3} d[n] \tag{2.50}$$

$$s[n] \xleftarrow{(5,3)} s[n] - \frac{1}{72}(-5d[n-1] - 24d[n] + 5d[n+1]) \tag{2.51}$$

$$s[n] \xleftarrow{(5,5)} s[n] - \frac{1}{2304} \begin{pmatrix} 35d[n-2] - 230d[n-1] - 768d[n] \\ +230d[n+1] - 35d\{n+2] \end{pmatrix} \tag{2.52}$$

Os fatores de normalização são:

$$n_L = 3\sqrt{2} \tag{2.53}$$

$$n_H = \frac{\sqrt{2}}{6} \tag{2.54}$$

2.6 CDF (6, x)

$$\tilde{g}_{(6,x)} : \frac{\sqrt{2}}{64} \cdot (1,-6,15,-20,15,-6,1) \tag{2.55}$$

$$\tilde{h}_{(6,2)} : \frac{\sqrt{2}}{64} \cdot (-5,30,-56,-14,154,-14,-56,30,-5) \tag{2.56}$$

$$\tilde{h}_{(6,4)} : \frac{\sqrt{2}}{2048} \cdot \begin{pmatrix} 35,-210,330,470,-1827,252,394, \\ 252,-1827,470,330,-210,35 \end{pmatrix} \tag{2.57}$$

$$\tilde{h}_{(6,6)} : \frac{\sqrt{2}}{16384} \cdot \begin{pmatrix} -63,378,-476,-1554,4404,1114,-13860,4158,28182, \\ 4158,-13860,1114,4404,-1554,-476,378,-63 \end{pmatrix} \tag{2.58}$$

Os passos *lifting* são:

$$d[n] \leftarrow d[n] - \frac{1}{6}(s[n] + s[n+1]) \tag{2.59}$$

$$s[n] \leftarrow s[n] - \frac{1}{16}(9d[n-1] + 9d[n]) \tag{2.60}$$

$$d[n] \leftarrow d[n] - \frac{1}{3}(4s[n] + 4s[n+1]) \tag{2.61}$$

$$s[n] \xleftarrow{(6,2)} s[n] - \frac{1}{32}(-5d[n-1] - 5d[n]) \tag{2.62}$$

$$s[n] \xleftarrow{(6,4)} s[n] - \frac{1}{1024}(35d[n-2] - 195d[n-1] - 195d[n] + 35d[n+1]) \tag{2.63}$$

$$s[n] \xleftarrow{(6,6)} s[n] - \frac{1}{8192}\begin{pmatrix} -63d[n-3] + 469d[n-2] - 1686d[n-1] \\ -1686d[n] + 469d[n+1] - 63d[n+2] \end{pmatrix} \tag{2.64}$$

Os fatores de normalização são:

$$n_L = 4\sqrt{2} \tag{2.65}$$

$$n_H = \frac{\sqrt{2}}{8} \tag{2.66}$$

2.7 Transformadas de Inteiros

Mostraremos a seguir algumas transformadas *wavelet* de inteiros reversíveis que foram apresentadas em [4, 5] e também usadas em [2, 56], com a notação $(N, \tilde{N})$, onde N e $\tilde{N}$ representam o número de momentos de anulação de análise é síntese dos filtros passa alta, respectivamente.

A transformada *wavelet* de inteiros, isto é, a transformada *wavelet* que mapeia inteiros para inteiros, pode ser desenvolvido usando o *lifting scheme* arredondando para o menor inteiro o resultado de cada passo *lifting* dual e primário antes da adição ou subtração. O passo *lifting* dual é calculado por

$$d^{(k)}[n] = d^{(k-1)}[n] - \left\lfloor (\sum_m p^{(k)}[m] s^{(k-1)}[n-m]) + \frac{1}{2} \right\rfloor \qquad (6.67)$$

e o passo *lifting* primário é calculado por

$$s^{(k)}[n] = s^{(k-1)}[n] - \left\lfloor (\sum_m u^{(k)}[m] d^{(k)}[n-m]) + \frac{1}{2} \right\rfloor \qquad (6.68)$$

A transformada inversa pode ser obtida aplicando os passos *lifting* dual e primário na ordem inversa em que foi aplicado a transformada direta e invertendo também os sinais,

$$s^{(k-1)}[n] = s^{(k)}[n] + \left\lfloor (\sum_m u^{(k)}[m] d^{(k)}[n-m]) + \frac{1}{2} \right\rfloor, \text{ e} \qquad (6.69)$$

$$d^{(k-1)}[n] = d^{(k)}[n] + \left\lfloor (\sum_m p^{(k)}[m] s^{(k-1)}[n-m]) + \frac{1}{2} \right\rfloor. \qquad (6.70)$$

Para todos os exemplos será considerado o fator de normalização $K = 1$. Apresentaremos apenas as transformadas diretas, já que a transformada inversa é simples de ser obtida conforme já mostrado acima.

- **Transformada (2, 2):**

$$d[n] = x[2n+1] - \left\lfloor \frac{1}{2}(x[2n] + x[2n+2]) + \frac{1}{2} \right\rfloor \quad (2.71)$$

$$s[n] = x[2n] + \left\lfloor \frac{1}{4}(d[n-1] + d[n]) + \frac{1}{2} \right\rfloor \quad (2.72)$$

- **Transformada (4, 2)**

$$d[n] = x[2n+1] - \left\lfloor \frac{9}{16}(x[2n] + x[2n+2]) - \frac{1}{16}(x[2n-2] + x[2n+4]) + \frac{1}{2} \right\rfloor \quad (2.73)$$

$$s[n] = x[2n] + \left\lfloor \frac{1}{4}(d[n-1] + d[n]) + \frac{1}{2} \right\rfloor \quad (2.74)$$

- **Transformada (2, 4)**

$$d[n] = x[2n+1] - \left\lfloor \frac{1}{2}(x[2n] + x[2n+2]) + \frac{1}{2} \right\rfloor \quad (2.75)$$

$$s[n] = x[2n] + \left\lfloor \frac{19}{64}(d[n-1] + d[n]) - \frac{3}{64}(d[n-2] + d[n+1]) + \frac{1}{2} \right\rfloor \quad (2.76)$$

- **Transformada (6, 2)**

$$d[n] = x[2n+1] - \left\lfloor \begin{array}{c} \frac{75}{128}(x[2n] + x[2n+2]) - \frac{25}{256}(x[2n-2] + x[2n+4]) \\ + \frac{3}{256}(x[2n-4] + x[2n+6]) + \frac{1}{2} \end{array} \right\rfloor \quad (2.77)$$

$$s[n] = x[2n] + \left\lfloor \frac{1}{4}(d[n-1] + d[n]) + \frac{1}{2} \right\rfloor \quad (2.78)$$

- **Transformada (4, 4)**

$$d[n] = x[2n+1] - \left\lfloor \frac{9}{16}(x[2n] + x[2n+2] - \frac{1}{16}(x[2n-2] + x[2n+4]) + \frac{1}{2} \right\rfloor \quad (2.79)$$

$$s[n] = x[2n] + \left\lfloor \frac{9}{32}(d[n-1] + d[n]) - \frac{1}{32}(d[n-2] + d[n+1]) + \frac{1}{2} \right\rfloor \quad (2.80)$$

- **Transformada (2, 2+2)**

$$d^{(1)}[n] = x[2n+1] - \left\lfloor \frac{1}{2}(x[2n] + x[2n+2]) + \frac{1}{2} \right\rfloor \quad (2.81)$$

$$s[n] = x[2n] + \left\lfloor \frac{1}{4}(d^{(1)}[n-1] + d^{(1)}[n]) + \frac{1}{2} \right\rfloor \quad (2.82)$$

$$d[n] = d^{(1)}[n] - \left\lfloor \frac{1}{16}(-s[n-1] + s[n] + s[n+1] - s[n+2]) + \frac{1}{2} \right\rfloor \quad (2.83)$$

As três transformadas de inteiros apresentadas na seção 4.2, ou seja, as transformadas S, S+P e TS, podem ser vistas como casos especiais do *lifting scheme*. Se utilizarmos a notação apresentada acima, a transformada S é referida como transformada (1, 1) e a transformada TS é uma versão inteira da transformada *wavelet* biortogonal (3, 1) de CDF.

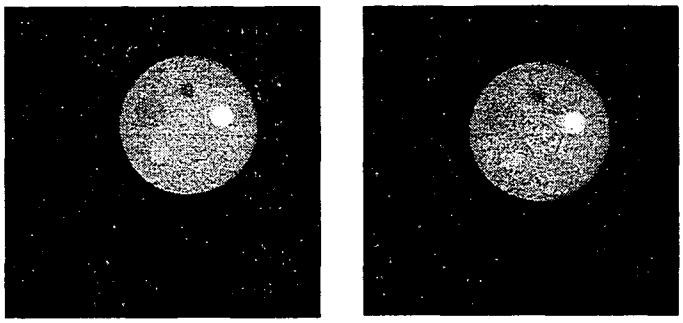
Apêndice 3 - Senogramas

As imagens seguintes correspondem aos senogramas, e suas respectivas imagens reconstruídas, que foram utilizadas na realização dos testes neste trabalho. O termo reconstrução usado aqui não se trata da aplicação do processo inverso utilizado para comprimir os dados, mas sim o processo que permite reconstruir a imagem de tomografia a partir das projeções.

Nessas imagens o preto corresponde ao *pixel* de valor 0 e o branco corresponde ao *pixel* de valor 65535. As imagens 3.1 e 3.3 (amostra 4 e amostra 10 respectivamente) correspondem ao senograma obtidos a partir do *phantom* de teste utilizando diferentes elementos no interior dos furos do objeto. As imagens 3.2 (a) e 3.4 são as imagens reconstruídas.

Apenas para fins ilustrativo, na imagem 3.2 (b) temos a reconstrução da imagem 3.1 com perdas. Aplicamos um limiar de 512 após a aplicação da transformada *wavelet*, inserindo assim uma pequena perda. Com a aplicação deste limiar, foi possível obter uma taxa média de 1,43 bits (taxa de compressão de aproximadamente 82%), porém é possível perceber a diferença entre as duas imagens reconstruídas.

Imagem 3.1 – Imagem am4.



(a) (b)

Imagem 3.2 – Imagem am4 após a reconstrução, (a) sem perdas e (b) com perdas.

Imagem 3.3 – Imagem amos_10.

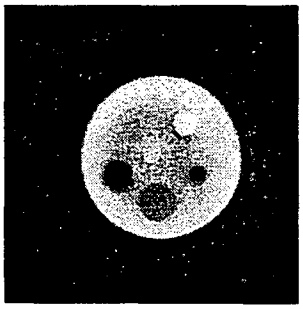


Imagem 3.4 – Imagem amos_10 após a reconstrução.

As duas imagens seguintes 3.5 e 3.7, correspondem a duas fatias diferentes obtidas de um fruto do babaçu. As imagens 3.6 e 3.8 são as imagens após a reconstrução.

Imagem 3.5 – Imagem babacu.

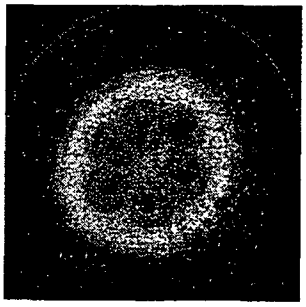


Imagem 3.6 – Imagem babacu após a reconstrução.

Imagem 3.7 – Imagem babacu2.

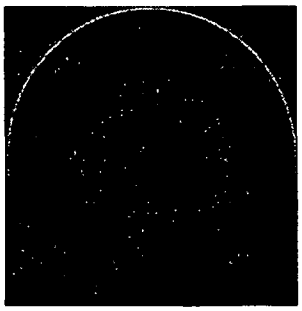


Imagem 3.8 – Imagem babacu2 após a reconstrução.

As imagens 3.9 e 3.10 correspondem ao senograma de um cabo elétrico de alta tensão e sua reconstrução respectivamente.

Imagem 3.9 – Imagem cabo2_3.

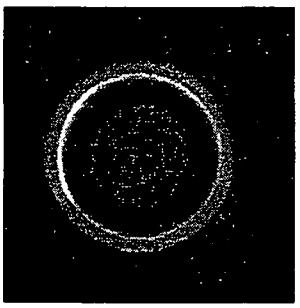


Imagem 3.10 – Imagem cabo2_3 após a reconstrução.

As imagens 3.11 e 3.13 correspondem a duas amostras de concreto, enquanto que as imagens 3.12 e 3.14 representam a reconstrução das amostras.

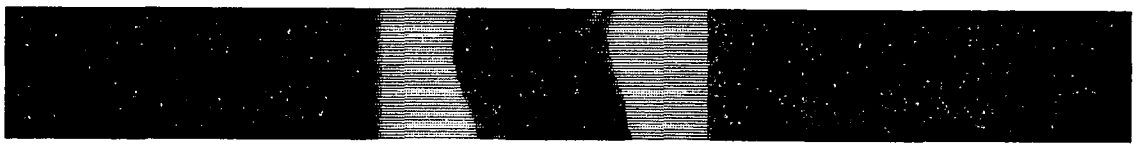


Imagem 3.11 – Imagem concret2.

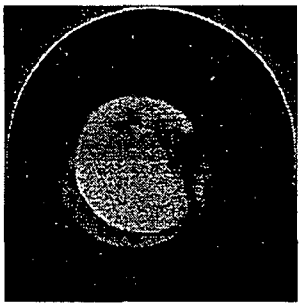


Imagem 3.12 – Imagem concret2 após a reconstrução.

Imagem 3.13 – Imagem concreto.

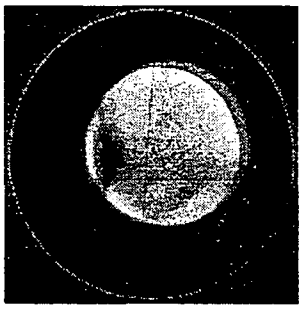


Imagem 3.14 – Imagem concreto após a reconstrução.

As demais imagens correspondem a três fatias diferentes de um dente humano. As imagens 3.15, 3.17 e 3.19 são os senogramas e as imagens 3.16, 3.18 e 3.20 as suas respectivas reconstruções.

Imagem 3.15 – Imagem dente01.

Imagem 3.16 – Imagem dente01 após a reconstrução.

Imagem 3.17 – Imagem dente02.

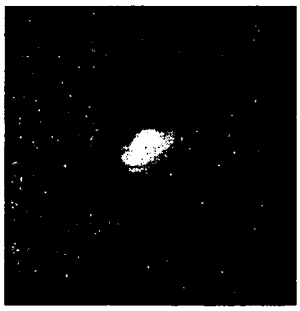


Imagem 3.18 – Imagem dente02 após a reconstrução.

Imagem 3.19 – Imagem dente03.

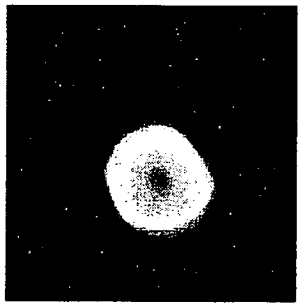


Imagem 3.20 – Imagem dente03 após a reconstrução.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] ANTONINI, M., BARLAUD, M., MATHIEU, P., DAUBECHIES, I., "Image Coding using Wavelet Transform", IEEE Transactions on Image Processing, v. 1, pp. 205-220, April, 1992.

[2] BILGIN, A., ZWEIG, G., MARCELLIN, M. W., "Efficient Lossless Coding of Medical Image Volumes Using Reversible Integer Wavelet Transforms", Proc. of 1998 Data Compression Conference, pp. 428-437, Snowbird, Utah, March, 1998.

[3] BRANNAN, P., "Implementation of a Lossless *Wavelet* Transform in C", C-Scene Issue, n. 2, August, 1997.

[4] CALDERBANK, R. C., DAUBECHIES, I., SWELDENS, W., e YEO, B., "Lossless Image Compression using Integer to Integer Wavelet Transforms", International Conference on Image Processing (ICIP), v. 1, p. 596-599, 1997.

[5] CALDERBANK, R. C., DAUBECHIES, I., SWELDENS, W., e YEO, B., "Wavelet Transforms that Map Integers to Integers", Department of Mathematics, Princeton University, 1996.

[6] CHAO, H., FISHER, P., "An Approach to Fast Integer Reversible Wavelet Transforms for Image Compression", Preprint, September, 1996.

[7] CHAO, H., FISHER, P., HUA, Z., "An Approach to Integer Wavelet Transformations for Lossless Image Compression", Preprint Revised, December, 1997.

[8] CODY, M. A., "The Fast Wavelet Transform Beyond Fourier Transforms", Dr. Dobb's Journal, v. 17, n. 4, April, 1992.

[9] COHEN, A., DAUBECHIES, I., FEAUVEAU, J., "Biorthogonal Bases of Compactly Supported Wavelets", Communications on Pure and Applied Mathematics, v. 45, pp. 485-560, 1992.

[10] COHEN, M. F., DeROSE, T. D., FOURNIER, A., LOUNSBERY, M., REISSEL, L-M., SCHRÖDER, P., SWELDENS, W., "Wavelets and their Applications in Computer Graphics", SIGGRAPH 95 Conference, Course Notes #26, ACM, 1995.

[11] DAUBECHIES, I., "Orthonormal Bases of Compactly Supported Wavelets", Communications on Pure and Applied Mathematics, v. 41, pp. 909-996, November, 1988.

[12] DAUBECHIES, I., SWELDENS, W., "Factoring Wavelet Transforms into Lifting Steps", Journal of Fourier Analysis and Applications, v. 4, n. 3, pp. 247-269, 1998.

[13] DENECKER, K., ASSCHE, S. V., PHILIPS, W., LEMAHIEU, I., "State of the Art Concerning Lossless Medical Image Coding", in Proc. of the PRORISC IEEE

Benelux Workshop on Circuits, Systems and Signal Processing, pp. 129-136, STW Technology Foundation, November, 1997.

[14] DENECKER, K., OVERLOOP, J. V., LEMAHIEU, I., "An Experimental Comparison of Several Lossless Image Coders for Medical Images", in Proc. of the Data Compression Industry Workshop, Snowbird, Utah, USA, pp. 67-76, Ball Aerospace & Technologies Corp., March, 1997.

[15] DeVORE, R. A., JAWERTH, B., LUCIER, B. J., "Image Compression Through Wavelet Transform Coding", IEEE transactions on Information Theory, v. 38, n. 2, pp. 719-746, March, 1992.

[16] DEWITTE, S., CORNELIS, J., "Lossless Integer Wavelet Transform", IEEE Signal Processing Letters, v. 4, n. 6, pp. 158-160, June, 1997.

[17] DOVICCHI, J. C. L., "Novos Coeficientes Wavelets Baseados em Intervalos Musicais para Análise de Timbres de Instrumentos Acústicos", Tese de Doutorado, Universidade Federal de Uberlândia, Uberlândia, 1999.

[18] EDWARDS, T., "Discrete Wavelet Transform: Theory and Implementation", *Tech. Rep.*, Stanford University, September, 1991.

[19] GALVÃO, R. K. H., RABELLO, T. N., Wavelets e Redes Neurais, V Escola de Redes Neurais, ITA, São José dos Campos, Julho, 1999.

[20] GOMES, J., VELHO, L., "Computação Gráfica: Imagem", Rio de Janeiro, IMPA/SBM, 1994.

[21] GONZALEZ, R. C., WOODS, R. E., "Digital Image Processing", Addison-Wesley Publishing Company, 1992.

[22] GRAPS, A., "An Introduction to Wavelets", IEEE Computational Science and Engineering, v. 2, n. 2, 1995.

[23] GROSSMAN, A., MORLET, J., "Decomposition of Hardy Functions into Square-Integrable Wavelets of Constant Shape", SIAM Journal of Math. Anal., v. 15, n. 4, pp. 723-736, July, 1984.

[24] HAAR, A., "Zur Theorie der Orthogonalen Funktionensysteme", Math. Annal., v. 69, pp. 331-371, 1910.

[25] HELD, G., "Compressão de Dados: Técnicas e Aplicações, Considerações de Hardware e Software", São Paulo, Érica, 1992.

[26] HERMAN, G. T., "Image Reconstruction from Projections", Academic Press Inc., Orlando, Florida, 1980.

[27] HOWARD, P. G., VITTER, J. S., "Fast and Efficient Lossless Image Compression", in Proceedings of the IEEE Data Compression Conference, Snowbird, Utah, pp. 351-360, April, 1993.

[28] JAIN, A. K., "Fundamentals of Digital Image Processing", Prentice Hall, 1989.

[29] JAWERTH, B., SWELDENS, W., "An Overview of Wavelet Based Multiresolution Analyses", SIAM Rev., v. 36, n. 3, pp. 377-412, 1994.

[30] KOVACEVIC, J., SWELDENS, W., "Wavelet Families of Increasing Order in Arbitrary Dimensions", Submitted to IEEE Transactions on Image Processing, 1997.

[31] LEWIS, A, S., KNOWLES, G., "Image Compression using the 2-D Wavelet Transform", IEEE Transactions on Image Processing, v. 1, n. 2, pp. 244-250, April, 1992.

[32] MALLAT, S. G., "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation", IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 11, n. 7, July, pp. 674-693, 1989.

[33] MANDUCA, A., SAID, A., "Wavelet Compression of Medical Images with Set Partitioning in Hierarchical Trees", SPIE Symposium on Medical Imaging, Cambridge, MA, March, 1996.

[34] MARCELLIN, M. W., GORMISH, M. J., BILGIN, A., BOLIEK, M. P., "An Overview of JPEG-2000", Proceedings of 2000 Data Compression Conference, Snowbird, March, 2000.

[35] MELO, A. C. W. P., "Compressão de Imagens em Escala de Cinza Utilizando a Transformada Wavelet", Dissertação de Mestrado, CEFET-PR, Curitiba, 1994.

[36] MOFFAT, A., WITTEN, I. H., NEAL, R. M., "Arithmetic Coding Revisited", *ACM Transactions on Information Systems*, v. 16, n. 3, p. 256-294, July, 1998.

[37] MULCAHY, C., "Image Compression Using the Haar Wavelet Transform", Spelman College Science and Mathematics Journal, v. 1, n. 1, pp. 22-31, April, 1997.

[38] NAYLOR, A. W., SELL, G. R., "Linear Operator Theory in Engineering and Science", Springer, New York, 1982.

[39] NELSON, M., GAILLY, J. L., "The Data Compression Book", M&T Books, 2nd Edition, New York, 1996.

[40] OLIVEIRA, L. F., "Aplicações da Transformada Wavelet em Compressão de Imagens", Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro, 1998.

[41] OLSON, T., "Optimal Time-Frequency Projections for Localized Tomography", Annals of Biomedical Engineering, v. 23, Issue 5, pp. 622-636, September/October, 1995.

[42] OPPENHEIM, A, V., SCHAFER, R, W., "Discrete-Time Signal Processing", Prentice-Hall Inc., Englewood Cliffs, NJ, 1989.

[43] ORTEGA, A., RAMCHANDRAN, K., "Rate-Distortion Methods for Image and Video Compression", IEEE Signal Processing Magazine, pp. 23-50, November, 1998.

[44] POLIKAR, R., "The Wavelet Tutorial", Iowa State University, Ames, IA, 1998.

[45] PRASAD, L., IYENGAR, S. S., "Wavelet Analysis with Applications to Image Processing", CRC Press, New York, 1997.

[46] RABBANI, M., "JPEG-2000: Background, Scope, and Technical Description", Eastman Kodak Company, Rochester, NY, December, 1998.

[47] RAMASWAMY, V. N., "Lossless Image Compression using *Wavelet* Decomposition", Ph.D. thesis, University of South Florida, Department of Computer Science and Engineering, Tampa, Florida, August, 1998.

[48] REID, M. M., MILLAR, R. J., BLACK, N. D., "Second-Generation Image Coding: An Overview", ACM Computing Surveys, v. 29, n. 1, March, 1997.

[49] RIBEIRO, E. P., "Tomografia de Susceptibilidade Magnética com Magnetômetro Supercondutor SQUID", Tese de Doutorado, PUC-RJ, Rio de Janeiro, Dezembro, 1996.

[50] SAID, A., PEARLMAN, W. A., "A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees", IEEE Transactions on Circuits and Systems for Video Technology, v. 6, n. 3, pp. 243-249, June, 1996.

[51] SAID, A., PEARLMAN, W. A., "An Image Multiresolution Representation for Lossless and Lossy Compression", IEEE Transactions on Image Processing, v. 5, n. 9, pp. 1303-1310, September, 1996.

[52] SAID, A., PEARLMAN, W. A., "Reversible Image Compression Via Multiresolution Representation and Predictive Coding", Proc. SPIE Conf. Visual Communications and Image Processing, v. 2094, pp. 664-674, Cambridge, November, 1993.

[53] SCHWARTZ, E. L., ZANDI, A., BOLIEK, M., "Implementation of Compression with Reversible Embedded Wavelets", Proceedings of SPIE, v. 2564, n. 4, 1995.

[54] SHANNON, C. E., "A mathematical theory of communication", *The Bell System Technical Journal*, v. 27, p. 379-423, 623-656, July, October, 1948.

[55] SHAPIRO, J. M., "Embedded Image Coding using Zerotrees of Wavelet Coefficients", IEEE Transactions on Signal Processing, v. 41, n. 12, pp. 3445-3462, December, 1993.

[56] SHENG, F., BILGIN, A., SEMENTILLI, P. J., MARCELLIN, M. W., "Lossy and Lossless Image Compression Using Reversible Integer Wavelet Transforms", Proc. of International Conference on Image Processing, Chicago, Illinois, October, 1998.

[57] STOLLNITZ, E. J., DeROSE, T. D., SALESIN, D. H., "Wavelets for Computer Graphics: A Primer, Part 1", IEEE Computer Graphics and Applications, v. 15, n. 3, pp. 76-84, May, 1995.

[58] STORER, J. A., "Data Compression: Methods and Theory", Computer Science Press, Inc., Rockville, MD, 1988.

[59] SWELDENS, W., "The Lifting Scheme: A Construction of Second Generation Wavelets", SIAM J. Math. Anal, v. 29, n. 2, pp. 511-546, 1997.

[60] SWELDENS, W., "The Lifting Scheme: A Custom-Design Construction of Biorthogonal Wavelets", Journal of Applied and Computational Harmonic Analysis, v. 3, n. 2, pp. 186-200, 1996.

[61] SWELDENS, W., "The Lifting Scheme: A New Philosophy in Biorthogonal Wavelet Constructions", Wavelet Applications in Signal and Image Processing III, pp. 68-79, Proc. SPIE 2569, 1995.

[62] SWELDENS, W., "Wavelets and the Lifting Scheme: A 5 Minute Tour", Zeitschrift für Angewandte Mathematik und Mechanik, v. 76 (Suppl. 2), pp. 41-44, 1996.

[63] SWELDENS, W., PIESSENS, R., "Wavelet Sampling Techniques", Proceedings of the Statistical Computing Section, pp. 20-29, American Statistical Association, 1993.

[64] UYTTERHOEVEN, G., "Wavelets: Software and Applications", Ph.D. thesis, Katholieke Universiteit Leuven, Department of Computer Science, Heverlee, Belgium, April, 1999.

[65] UYTTERHOEVEN, G., ROOSE, D., BULTHEEL, A., "Wavelet transforms using the Lifting Scheme", ITA-Wavelets Report WP 1.1, Department of Computer Science, Katholieke Universiteit Leuven, Belgium, April, 1997.

[66] VILLASENOR, J. D., BELZER, B., LIAO, J., "Wavelet Filter Evaluation for Image Compression", IEEE Transactions on Image Processing, v. 2, pp. 1053-1060, August, 1995.

[67] WALLACE, G. K., "The JPEG Still Picture Compression Standard", Communications of the ACM, v. 34, n. 4, p. 30-44, April, 1991.

[68] WEINBERG, M. J., SEROUSSI, G., SAPIRO, G., "LOCO-I: A Low Complexity, Context-Based, Lossless Image Compression Algorithm", in Proceedings of IEEE Data Compression Conference, Snowbird, Utah, pp. 140-149, March-April, 1996.

[69] WELCH, T. A., "A Technique for High-Performance Data Compression", IEEE Computer, v. 17, n. 6, p. 8-19, June, 1984

[70] WITTEN, I. H., NEAL, R. M., CLEARY, J. G., "Arithmetic Coding for Data Compression", *Communications of the ACM*, v. 30, n. 6, p. 520-540, June, 1987.

[71] WU, X., MEMON, N., SAYOOD, K., "A Context-Based Adaptive, Lossless/Nearly-Lossless Coding Scheme for Continuous-Tone Images", Proposal submitted for a initial ISO/IEC JTC 1.29.12 evaluation, July, 1995.

[72] ZANDI, A., ALLEN, J. D., SCHWARTZ, E. L., BOLIEK, M., "CREW: Compression with Reversible Embedded Wavelets", Proceedings of IEEE Data Compression Conference, Snowbird, Utah, pp. 212-221, March, 1995.

[73] ZIV, J., LEMPEL, A., "A Universal Algorithm for Sequential Data Compression", IEEE Transactions on Information Theory, v. IT-23, n. 3, p. 337-343, May, 1977.

[74] ZIV, J., LEMPEL, A., "Compression of Individual Sequences via Variable-Rate Coding", *IEEE Transactions Information Theory*, v. IT-24, p. 530-536, September, 1978.