UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E COMPUTAÇÃO
DEPARTAMENTO DE COMUNICAÇÕES

PROCESSAMENTO E COMPRESSÃO DO SINAL DE VÍDEO UTILIZANDO A TRANSFORMADA *WAVELET*

Vicente Idalberto Becerra Sablón

Orientador: Prof. Dr. Yuzo Iano

Banca Examinadora:
Prof. Dr. Ayres Mardem Almeida do Nascimento (UFMA)
Prof. Dr. César Kyn D'Ávila (CEDET)
Prof. Dr. Dalton Soares Arantes (FEEC – UNICAMP)
Prof. Dr. João Baptista Tadanobu Yabu-uti (FEEC – UNICAMP)
Prof. Dr. Edson Moschim (FEEC – UNICAMP)

Tese apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos exigidos para a obtenção do titulo de Doutor em Engenharia Elétrica

Campinas-SP-Brasil
Dezembro de 2002

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

B386p

Becerra Sablón, Vicente Idalberto
Processamento e compressão do sinal de vídeo utilizando transformada wavelet / Vicente Idalberto Becerra Sablón. --Campinas, SP: [s.n.], 2002.

Orientador: Iano, Yuzo.
Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Processamento de imagem – Técnicas digitais. 2. Compressão de dados (Telecomunicações). 3. Compressão de imagens. 4. Processamento de sinais. I. Iano,Yuzo. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

AGRADECIMENTOS

Aos meus filhos ***Sandra e Daniel*** e a minha esposa ***Zaida*** pelo carinho, amor e companhia em horas alegres e dificeis.

Ao meu orientador ***Prof. Dr.Yuzo Iano*** por incontáveis horas dedicadas à minha orientação, pelo estímulo, consideração e amizade que foram essenciais na realização deste trabalho.

Agradeço de **modo especial** à ***Fundação de Amparo à Pesquisa do Estado de São Paulo*** (***FAPESP***) que durante todos esses anos forneceu suporte financeiro para a execução deste trabalho através de bolsas e de reservas técnicas. Agradeço também ao ***Conselho Nacional de Desenvolvimento Científico e Tecnológico*** (***CNPq***), à **Coordenação de Aperfeiçoamento de pessoal de Nível Superior** (***CAPES***) pelo apoio financeiro de forma geral.

A ***Faculdade de Engenharia Elétrica da UNICAMP***, por ter me aceito na sua pós graduação.

Ao governo *da* ***Republica Federativa do Brasil*** e ao **povo brasileiro** por dar-me a oportunidade de reunificar a minha família neste acolhedor pais.

Aos ***meus pais e irmão*** pelo carinho e atenção que dispensaram durante toda a minha vida.

Aos ***meus amigos, Fabio e Ilva,*** que acolheram-me na sua família tornando agradável a vida longe do meu pais e propiciando-me tranqüilidade e afeto nas horas difíceis.

A minha ***amiga Angela*** pela ajuda, companhia e carinho dedicados aos meus filhos.

Aos ***meus amigos do departamento Alessandra, Ahmed, Hadi, Antônio, Ayres, Guille, Wilson, Ana, Fernando, Jorge, Nalon, Felipe e Marcelo***, pela companhia, amizade e apoio que me brindaram em tudo que foi necessário.

Aos ***meus amigos Luis e Fabbryccio*** da ***TCP-TELECOM***, pela dedicação, apoio e incentivo dados para a finalização deste trabalho.

Aos amigos cubanos ***Eduardo, Martha, Liset, Frank, Oselys , Victor e Maximino*** por me ajudar e propiciar a minha família carinho e solidariedade nestes anos.

Aos ***professores da Feec/Unicamp. Prof. Dr. Dalton Soares Arantes, Prof Dr. José Geraldo Chiquito, Prof Dr. João Baptista Tadanobu Yabu-uti, Prof Dr. Edson Moschim*** pela clareza e simpatia com que transmitiram-me seus conhecimentos, fosse em aulas ou em conversas particulares.

Aos ***funcionários , Suely, Lúcia*** meus agradecimentos pela ajuda.

Aos ***professores da banca examinadora*** e ao **Dr. César Kyn D'Ávila** agradeço a presença e a participação.

A todos aqueles que contribuíram de alguma forma para a realização deste trabalho, meus sinceros agradecimentos.

Vicente Becerra Sablon

Campinas, Dezembro 2002

RESUMO

Neste trabalho apresenta-se um estudo da aplicabilidade da transformada "wavelet", no processamento e na compressão de sinais de vídeo. O estudo é realizado como uma conseqüência dos bons resultados obtidos na aplicação dessa transformada na compressão e processamento de imagens estáticas. Foi realizado também, um estudo sobre as principais técnicas de compressão de imagens, enfatizando-se os codificadores com perdas. Apresentou-se os conceitos teóricos relacionados com a transformada em questão e foram feitas simulações para se avaliar as ferramentas matemáticas disponíveis, bem como para a avaliação do desempenho, compatibilidade e eficiência dos esquemas propostos para a compressão de sinais de vídeo. Propõe-se um método de estimação de movimento no domínio wavelet a partir da técnica de codificação PACC (Codificação Condicional Particionada e Agregada) conjuntamente com a propriedade da transformada ser variante com o deslocamento.

ABSTRACT

A study of the applicability of the wavelet transformed, in the processing and in video signals compression is presented in this work. The study was carried out as a consequence of the good results obtained in the application of that transform in the compression and processing of static images. A study on the main techniques of images compression, with emphasis on encoders with losses and theoretical concepts related with transformed images is also presented. Simulations were done in order to evaluate the available mathematical tools for that purposes, as well as for the evaluation of the performance, compatibility and efficiency of the outlines proposed for the compression of video signals. A method of movement estimation in the wavelet domain coming from the coding technique PACC (Conditional Code Partitioning and Joined) jointly with the property of the transform to be shift variant is presented as a new proposal.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE ABREVIATURAS.

$A_{2^{j+1}} f(x)$: A aproximação da função $f(x)$ no nível de resolução 2^{j+1}

$A_{2^j}^d f$: aproximação discreta do sinal $f(x)$ no nível de resolução 2^j.

A_{2^j} : operador que gera uma aproximação com resolução 2^j de um sinal

$\psi(x)$: *wavelet*-mãe

$\|\psi\| = 1$: *wavelet*-mãe de norma unitária

$\psi_{j,k}(x)$: *wavelets*-filhotes

$\phi(x)$: função escala

$D_{2^j}^d f$: O sinal de detalhamento de $f(x)$

V_{2^j} :conjunto de todas as possíveis aproximações das funções $L^2(R)$ no nível de resolução 2^j.

T: operador *frame*

$\Psi(\omega_x, \omega_y)$: transformada de Fourier de $\psi(x, y)$.

$\Psi_{a;b_x,b_y}(x, y)$: *wavelets*-filhotes geradas a partir da *wavelet*-mãe $\psi(x, y)$

$\left(V_{2^j}^2\right)_{j \in Z}$: seqüência de subespaços em $L^2(R^2)$.

FFT: Fast Fourier Transform

FIR: resposta impulsiva finita.

PACC: Partition, Agregation and Condition Coding.

TW: Transformada Wavelet

TWC: Transformada Wavelet contínua.

TWD: Transformada Wavelet Discreta.

$A_{2^j}^d f$: Imagem corresponde às freqüências mais baixas

$D_{2^j}^1 f$: Imagem corresponde às freqüências verticais altas

$D_{2^j}^2 f$: Imagem corresponde às freqüências horizontais altas

$D_{2^j}^3 f$: Imagem corresponde às freqüências altas em ambas às direções.

LH : subbanda que contêm os coeficientes de alta freqüência na vertical.

HL : subbanda que contêm os coeficientes de alta freqüência na horizontal.

HH : subbanda que contêm os coeficientes de alta freqüência na diagonal.

PD: densidades de probabilidades.

FS: busca total

EZW: *Embeded Zerotree Wavelet Coder*

SPIHT: *Set Partitioning in Hierarchical Trees*

MRWD: *Morphological Representation of Wavelet Data*

PPC: *Predictive Pyramid Coder*

SQS: *Self-Quantization of Subtrees*

WFC: *Wavelet-Fractal Coder*

CAPÍTULO 1.

INTRODUÇÃO

As pesquisas envolvendo *wavelets* proporcionaram na última década o desenvolvimento de importantes estudos realizados por pesquisadores das mais diferentes áreas. Talvez tenha sido um dos assuntos científicos que mais interessados tenha reunido, podendo-se citar algumas áreas importantes envolvidas, tais como processamento de sinais, estatística, equações diferenciais, análise numérica, geofísica, física quântica, ciência da computação, entre outras. Essa ampla e crescente demanda de aplicações se deve ao fato de que as propriedades da transformada *wavelets* mostrou para matemáticos, engenheiros, e pesquisadores em geral uma nova e elegante maneira de se ver o mundo (*world-view*).

A análise de Fourier é considerada uma poderosa ferramenta de análise espectral, não obstante existam problemas em que a solução através da transformada de Fourier, torna-se inadequada ou tem uma alta complexidade computacional. Um caso específico é a análise tempo-freqüência de sinais. A transformada *wavelet* de uma função (sinal) capta a informação da localização tempo-freqüência do sinal, ao contrário da transformada de Fourier que sacrifica a localização num domínio para garantir o domínio complementar. A propriedade de localização tempo-freqüência facilita consideravelmente o estudo do comportamento de sinais, tais como singularidade e suavidade, bem como das características das mudanças locais sem alterações significativas no estado do sinal quando essas características são comparadas com aquelas em outras regiões de freqüência ou tempo.

Além disso, no caso discreto, os coeficientes das séries *wavelets* de um sinal com características completamente localizadas no espaço tempo-freqüência, correspondem aos coeficientes com uma janela discreta de tempo-freqüência. Essa característica inerente à transformada *wavelet* discreta resulta na propriedade de multiresolução que possibilita o estudo do sinal com variação da resolução. Embora a transformada *wavelet* não substitua a transformada de Fourier, existem aplicações onde a transformada *wavelet* constitui-se em uma ferramenta "natural" relativamente superior para se efetuar a análise. Por exemplo, no processamento de sinais não estacionários, como é em geral, o caso das imagens, a

transformada *wavelet* não introduz efeitos de blocos. Sendo também a complexidade computacional inferior a da FFT -*Fast Fourier Transform*.

Este trabalho visa dar uma contribuição ao estudo de técnicas envolvendo o uso da transformada wavelet na compressão de vídeo.

O objetivo central é o de apresentar uma análise sobre as potencialidades de aplicação da transformada *wavelet* na compressão e processamento do sinal de vídeo. Realiza-se um estudo da teoria *wavelet*, bem como um estudo comparativo entre os algoritmos *wavelets* considerados como estado da arte em compressão de imagens. Analisa-se os principais conceitos e fundamentos associados a esses algoritmos que são considerados atuais para uso em compressão de imagem. Realiza-se também simulações para se avaliar tanto as ferramentas matemáticas disponíveis quanto a teoria estudada. Estuda-se também alguns esquemas de compressão de sinais de vídeo baseados na transformada *wavelet*, apresentando-se conceitos importantes tais como a escalonabilidade. Realizam-se simulações para se avaliar, o desempenho, a compatibilidade e a eficiência de esquemas de compressão de vídeo. Em seguida simulam-se os algoritmos baseados na transformada *wavelets* para compressão de imagens estáticas e estuda-se a sua aplicabilidade nos esquemas de compressão de vídeo. Como resultado final deste estudo propõe-se um esquema que permite realizar a estimação e compensação de movimentos no domínio *wavelet*.

O presente trabalho está estruturado da seguinte forma:

CAPITULO 2. *WAVELETS*.

Nesse capítulo apresenta-se um estudo da teoria de *wavelets* sob o ponto de vista da análise de multiresolução. A seguir definem-se as transformadas *wavelets* contínua e discreta e faz-se uma revisão da teoria de "*frames*". Um estudo da análise de multiresolução é realizado. Em seguida são apresentadas as *wavelets* biortogonais. Realiza-se então uma comparação entre o desempenho das *wavelets* ortogonais e biortogonais.

CAPITULO 3. TRANSFORMADA *WAVELET* NO PROCESSAMENTO DE IMAGENS

Nesse capítulo apresentam-se modelos de codificadores com perdas baseados na transformada *wavelet*. É realizado um estudo comparativo entre os algoritmos *wavelets* considerados como estado da arte em compressão de imagem. Através de simulações, analisam-se modelos em função do tipo de *wavelet*.

CAPITULO 4. SUBSISTEMA CODIFICADOR DE IMAGEM BASEADO NA TRANSFORMADA *WAVELET* PARA A CODIFICAÇÃO DOS SINAIS DE VÍDEO DIGITAL.

Nesse capítulo desenvolve-se um sistema que permite o processamento, a codificação e a decodificação de imagens estáticas aplicando-se a transformada *wavelet*. Esse sistema foi incluído como um bloco no esquema final do codificador de vídeo.

CAPITULO 5. CODIFICAÇÃO DE VÍDEO USANDO TÉCNICAS BASEADAS NA TRANSFORMADA *WAVELET* NO PRÉ-PROCESSAMENTO DO SINAL

Como conseqüência dos bons resultados obtidos na compressão de imagens estáticas, nesse capítulo apresenta-se a transformada *wavelet* como uma ferramenta para uso na área de compressão de imagens de vídeo. A transformada *wavelet* proporciona uma descorrelação eficiente da informação contida em cada quadro da seqüência, assim como acontece no caso das imagens estáticas. O objetivo central desse capítulo foi a implementação e o desenvolvimento de um sistema que permite o pré-processamento, codificação e decodificação de uma seqüência de imagens aplicando-se a transformada *wavelet*, em conjunto com outras técnicas de compressão, que foram estudadas e ajustadas para a realização dessa aplicação.

CAPITULO 6. ESTIMAÇÃO DE MOVIMENTO NO DOMÍNIO *WAVELET* A PARTIR DA PROPRIEDADE VARIANTE COM O DESLOCAMENTO

Nesse capítulo apresenta-se a técnica de estimação e compensação de movimento no domínio espacial e no domínio *wavelet*. A seguir se propõe-se um esquema de estimação e compensação de movimentos que permite dominar a propriedade de variância com o deslocamento (*"shift variant"*) da transformada *wavelet* discreta (TWD). Em seguida, são apresentados estudos comparativos de algumas técnicas de estimação e compensação de movimentos para a codificação de sinais de vídeo baseada na TW.

CAPITULO 7: SIMULAÇÕES E RESULTADOS

Nesse capítulo mostram-se os testes e resultados parciais e totais realizados em cada um dos sistemas, assim como os fundamentos que levaram à proposta do esquema final onde se propõe uma técnica para a estimação de movimento no domínio *wavelet* a partir da propriedade da variância com o deslocamento, juntamente com o modelo de pré-processamento conhecido como PACC (*"Partition, Agregation and Condition Coding"*).

CAPITULO 8. CONCLUSÕES E SUGESTÕES PARA TRABALHOS FUTUROS

São apresentados nesse capítulo os comentários finais sobre o trabalho e os resultados obtidos.

CAPÍTULO 2.

2.1. *WAVELETS*

I. INTRODUÇÃO

Analogamente à transformada de Fourier (TF), a transformada *wavelets* (TW) decompõe o sinal em conjuntos apropriados de bases de funções. *Wavelets* literalmente significam "*small waves*", "*ondelletes*" (pequenas ondas). Como o nome sugere, elas são ondas, oscilam e suas curvas têm um decaimento a zero em relação à área algébrica: $\int_{-\infty}^{\infty}\psi(t)dt = 0$.

O termo que sugere pequenez refere-se ao fato de que elas são localizadas no tempo. Isso contrasta com o comportamento das bases de Fourier, pois essas bases são constituídas por senos e cossenos infinitos. Essas funções senoidais e cossenoidais estão perfeitamente localizadas no espaço de freqüência, mas não decrescem a zero como uma função do tempo ("*nonlocal support*"). Por outro lado, as *wavelets* caem a zero quando $t \to \pm\infty$ e desfrutam da boa propriedade de localização no espaço de tempo.

A construção de uma *wavelet* é realizada a partir da função básica oscilatória de suporte compacto, denominada de *wavelet*-mãe (base), $\psi(t)$,que gera as bases a partir de deslocamentos e dilações $\psi(at-b)$. Para *wavelets* discretas, os parâmetros através dos quais acontece a dilação, a, e a translação, b, estão restritos a um conjunto discreto, usualmente, $a = 2^j$, $b = k$ onde j e k são inteiros. Essa estrutura permite que a TW ofereça uma excelente resolução em tempo e freqüência, permitindo a sua utilização no processamento de sinais não-estacionários, como, por exemplo, as imagens. Outra conseqüência importante da dilação é a possibilidade de uma representação hierárquica do conjunto de dados. Essa abordagem é também chamada de análise de multiresolução.

Ainda outra característica da TW é sua alta capacidade de concentrar a energia do sinal em um número reduzido de coeficientes, possibilitando a obtenção de uma representação mais compacta. A complexidade computacional da TW é inferior à da FFT ("*Fast Fourier Transform*"), o que motiva a sua utilização em sistemas práticos. A TW apresenta também

a vantagem de não introduzir os efeitos de blocagem. O conjunto dessas características fazem dessa transformada uma excelente alternativa aos métodos tradicionais de compressão de imagens.

II. TRANSFORMADA *WAVELET*

As mesmas três possibilidades que existem para a transformada de Fourier (TF), existem para a transformada *wavelet* (TW):

- A transformada *wavelet* contínua (TWC),
- Expansão em série de *wavelet*,
- A transformada *wavelet* discreta (TWD).

A situação é ligeiramente mais complexa do que a da TF, pois as funções bases da *wavelet* podem ou não ser ortonormais [1].

Os conjuntos das funções bases dessas *wavelets* podem sustentar a transformada até mesmo quando as funções não forem ortonormais. Isso significa, por exemplo, que uma expansão em série de *wavelet*, pode representar uma função limitada em banda, usando-se uma infinita quantidade de coeficientes. Se essa seqüência de coeficientes é truncada para se obter um comprimento finito, então pode-se reconstruir apenas uma aproximação da função original. Da mesma forma, a TWD pode requerer mais coeficientes do que pontos de amostras obtidas do sinal original a fim de se ter uma reconstrução exata, ou até mesmo uma aproximação aceitável [1].

II.1. TRANSFORMADA *WAVELET* CONTÍNUA

A transformada *wavelet* continua (TWC), foi introduzida por Grossman and Morlet [2].

Definição 2.1: Se $\psi(x)$ é uma função de valor real cuja TF satisfaz a *condição de admissibilidade* [2, 3]

$$C_\psi = \int_{-\infty}^{\infty} \frac{|\Psi(\omega)|^2}{|\omega|} d\omega < \infty, \tag{2.1}$$

onde C_ψ é uma constante e $\Psi(\omega)$ é a TF da *wavelet*-mãe $\psi(x)$, então essa condição de admissibilidade limita a classe de funções que podem ser utilizadas como *wavelet*-mãe. Além dessa restrição, $\psi(x)$ é uma função do tipo janela, ou seja,

$$\int_{-\infty}^{\infty}|\psi(x)|dx < \infty, \tag{2.2}$$

é contínua em $\boldsymbol{R}$ (conjunto dos números reais) [4, 5].

Da expressão (2.1), vemos que $\Psi(\omega)$ deve se anular na origem. Assim, dado que:

$$\Psi(\omega) = \int_{-\infty}^{\infty}\psi(x)e^{-2\pi f x}dx, \tag{2.3}$$

a exigência:

$$\Psi(0) = 0 \Rightarrow \int_{-\infty}^{\infty}\psi(x)dx = 0, \tag{2.4}$$

o que significa que $\psi(x)$ deve ser oscilante e de média nula.

De forma resumida pode-se dizer que a *wavelet* é ma função oscilante com média nula e que tem um decaimento rápido para zero.

O conjunto das *wavelets*-filhotes, $\psi_{a,b}(x)$ é constituído de versões deslocadas e dilatadas ou comprimidas da *wavelet*-mãe, $\psi(x)$ e que são capazes de gerar todo o espaço $L^2(R)$ (espaço vetorial unidimensional de funções mensuráveis, quadraticamente integráveis)

$$\psi_{a,b}(x) = \frac{1}{\sqrt{a}}\psi\left(\frac{x-b}{a}\right) \tag{2.5}$$

Na expressão (2.5), b é o parâmetro responsável pelo deslocamento da *wavelet* e a corresponde ao escalonamento, parâmetro responsável pela dilatação ou compressão. A constante $a^{-\frac{1}{2}}$ é o termo de normalização da energia da função em relação ao parâmetro **a**. Observe que ao aumentar-se o valor de **a**, a *wavelet* é dilatada e sua duração aumenta. Ao diminuir-se o valor de a, comprime-se a *wavelet* e sua duração diminui. Para valores negativos de b, a *wavelet* é deslocada para a esquerda e para valores positivos para a direita.

A transformada *wavelet* contínua (TWC) da função $f(x)$, é definida pela seguinte expressão [2, 3]:

$$TWC(a,b) = \left\langle f(x), \psi_{a,b}(x) \right\rangle = \int_{-\infty}^{\infty} f(x)\psi^{*}_{a,b}(x)dt \tag{2.6}$$

Para *se obter* a transformada *wavelet* inversa, isto é, recuperar $f(x)$ a partir dos seus coeficientes *wavelets* é necessário que a condição de admissibilidade seja satisfeita [4, 6].

Grossman e Morlet [2] demonstram, que se a constante $\bar{C}_{\psi}$ em (2.1) é finita, então, existe uma TW inversa, que é definida por

$$f(x) = \frac{1}{C_{\psi}} \iint_{R^2} TWC(a,b)\psi_{a,b}(x)\frac{dbda}{a^2}, \tag{2.7}$$

onde $f(x) \in \boldsymbol{L}^2(\boldsymbol{R})$ e $\boldsymbol{R}^2 = \boldsymbol{R} \times \boldsymbol{R}$.

II.1.1 A TWC COMO UM BANCO DE FILTROS

A Eq. (2.5) pode ser expressa da seguinte forma

$$\psi_a(x) = \frac{1}{\sqrt{a}}\psi\left(\frac{x}{a}\right), \tag{2.8}$$

onde $\psi_a(x)$ pode ser interpretada como uma *wavelet*-mãe escalonada por a e normalizada por $a^{-\frac{1}{2}}$.

Definindo-se

$$\breve{\psi}_a(x) = \psi_a{}^{*}(-x) = \frac{1}{\sqrt{a}}\psi^{*}(-\frac{x}{a}), \tag{2.9}$$

que é o complexo conjugado refletido da *wavelet*-mãe escalonada. Se $\psi(x)$ é real e par, como freqüentemente acontece, a reflexão e a conjugação não tem efeito. Pode-se escrever a TWC como

$$TWC(a,b) = \int_{-\infty}^{\infty} f(x)\breve{\psi}_a(b-x)dx = f(b) * \breve{\psi}_a(b), \tag{2.10}$$

Para um valor fixo de **a**, a *TWC (a,b)* é a convolução do sinal $f(x)$ com o conjugado refletido da *wavelet* – mãe na escala a, ou seja, a TWC de um sinal $f(x)$ corresponde à filtragem desse através de um filtro linear com resposta impulsiva $\breve{\psi}_a(t)$. A resposta em freqüência desse filtro é dada por

$$\breve{\Psi}_a(f) = \sqrt{a}\,\breve{\Psi}(af), \tag{2.11}$$

O valor da integral da magnitude quadrática de tais respostas é constante em relação ao parâmetro a. O banco de filtros da TWC é caracterizado por apresentar uma largura de faixa relativamente constante. Tais filtros são denominados de filtros com fator Q constante [7, 8].

Reduzindo-se o valor de a em (2.11), desloca-se a faixa do filtro para freqüências superiores. Como conseqüência, devido a desigualdade ou principio de incerteza de Heisenberg (Eq. 2.12), Δf aumenta e Δt diminui [9]. Para a análise de altas freqüências pode-se escolher valores pequenos de **a** e com isso obter-se uma boa localização no tempo.

$$\Delta f \times \Delta t \geq \frac{1}{4\pi} \tag{2.12}$$

A escala utilizada na transformada *wavelet* tem o mesmo significado da escala utilizada nos mapas geográficos, onde as escalas maiores fornecem visões globais enquanto que as menores nos permitem observar os detalhes. Na transformada *wavelet*, escalas grandes permitem analisar os trechos longos do sinal, enquanto que escalas menores aqueles trechos mais curtos de forma mais detalhada.

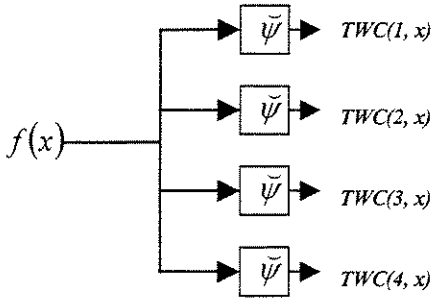


Fig. 2.1: Analogia de um banco de filtros obtidos a partir da TWC.

Na Fig. 2.1 mostra-se a TWC como um banco de filtros linear (convolução), atuando sobre $f(x)$. Cada valor de a define diferentes filtros passa-banda, e as saídas de todos eles juntas constituem a TWC. A Eq. 2.13,

$$f(x) = \frac{1}{C_\psi}\int_0^\infty\int_{-\infty}^{\infty}\left[f * \breve{\psi}_a\right](b)\,\psi_a(b-x)\frac{dbda}{a^2} = \frac{1}{C_\psi}\int_0^\infty\left[f * \breve{\psi}_a * \psi_a\right](x)\frac{da}{a^2}, \tag{2.13}$$

mostra, que para a reconstrução de $f(x)$, a saída dos filtros é novamente filtrada por $\psi_a(x)$ sendo adequadamente escalonada.

III. TRANSFORMADA *WAVELET* DISCRETA

Para se obter a transformada *wavelet* discreta (TWD), os parâmetros de deslocamentos e escalonamento são discretizados. Já a variável independente, isto é, o tempo, permanece contínua.

O conjunto de coeficientes que nos fornece a TWD, correspondem a pontos em uma grade bidimensional no domínio deslocamento-escala. Essa grade é indexada por dois números inteiros, *m* e *n*. O inteiro, *m* está relacionado à escala e o segundo inteiro, *n*, está relacionado ao deslocamento. O parâmetro de escalonamento *a* é discretizado de forma exponencial, $a = a_0^m$, enquanto que o parâmetro *b* é discretizado proporcional a *a*, $b = n.b_0.a_0^m$. As constantes a_0 *e* b_0 são os comprimentos dos passos discretos de escalonamento e de deslocamento, respectivamente. A TWD de um sinal $f(x)$ é definida como sendo [6, 9]

$$TWD(m,n) = \frac{1}{\sqrt{a_0^m}} \int_{-\infty}^{\infty} f(x)\psi^*(a_0^{-m}x - nb_0)dx \qquad (2.14)$$

Ao contrário da TWC, a TWD só está definida para valores positivos de escalonamento, ou seja, para $a_0 > 0$. Isso não chega a ser uma condição restritiva, já que pode-se utilizar uma *wavelet* refletida para cobrir as escalas negativas.

Para valores grandes de *a*, a resolução no tempo é pequena e conseqüentemente, os passos de deslocamentos são grandes. Para valores pequenos de a, a resolução no tempo é grande e os passos de deslocamentos são pequenos. Isso justifica o fato do passo de deslocamento ser proporcional ao escalonamento.

Como é ilustrado na Fig. 2.2, cada ponto corresponde-se com uma *wavelet* $\psi_{a,b}(x)$, para diferentes valores de *m*, correspondem *wavelets* de diferentes larguras. Por exemplo: *Estreitas* (*wavelets* de alta – freqüência), são deslocadas através de passos menores, para cobrir o eixo inteiro, enquanto as mais *largas* (*wavelets* de baixa – freqüência), são deslocadas através de passos maiores. A amplitude de um coeficiente para um dado

intervalo de tempo diminui exponencialmente. Conseqüentemente, o número de passos necessários para cobrir o intervalo diminui, já que o tamanho da janela aumenta.

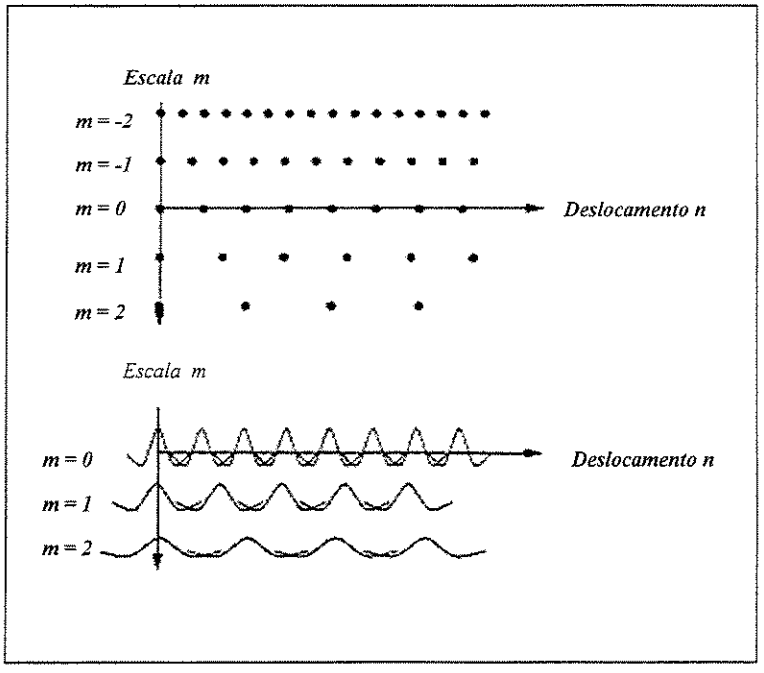


Fig. 2.2: Grade de amostragem e seu correspondente conjunto de funções, obtidas a partir da discretização dos parâmetros $a = a_0^m$ e $b = nb_0.a_0^m$. Neste exemplo $a_0 = 2^{1/2}, b_0 = 1$.

Definindo-se

$$a = a_0^j \quad \text{e} \quad b = nb_0a_0^j = ka_0^j, \quad j,k \in Z, \tag{2.15}$$

pode-se reescrever a expressão (2.14) da seguinte forma

$$TWD(j,k) = a_0^{-j/2} \int_{-\infty}^{\infty} f(x)\psi^*(a_0^{-j}x - k)dx \tag{2.16}$$

Pode-se comparar a análise de um sinal através das *wavelets* com a análise de um material em um microscópio. Para se analisar um material, a primeira providência que se deve tomar é se mover para o local escolhido, o que seria equivalente na análise com *wavelets* a se escolher o valor para k (Eq. 2.17). Em seguida, deve-se escolher o fator de amplificação, ou seja, a_0^{-j}. Desejando-se ver detalhes muito pequenos do material, a ampliação deve ser grande, o que corresponde a $j < 0$. Como conseqüência, o valor do produto ka_0^j na Eq.(2.15) será pequeno produzindo-se deslocamentos pequenos. Querendo-se visualizar uma área maior, deve-se escolher uma ampliação menor e, portanto, ka_0^j será maior, levando a deslocamentos maiores.

Para $a_0 = 2, b_0 = 1$, obtem-se a família das *wavelet* diádicas, que serão tratadas na próxima seção.

III.1. *WAVELETS* DIÁDICAS

Freqüentemente, o tamanho do passo da escala discreta é escolhido igual a 2, ou seja, tem-se $a_0 = 2$. Esse valor facilita a implementação, já que dilatar um sinal por um fator de 2 corresponde simplesmente a tomar uma amostra sim e outra não do sinal [10]. Na Fig. 2.2, $a_0 = 2$ e cada vez que se aumenta **a** por um fator de 2 o número de coeficientes diminui também por um fator de 2. Assim, quando $a_0 = 2$, diz-se que as freqüências são particionadas em oitavas consecutivas.

Se uma única *wavelet*-mãe, $\psi(x)$, permite a obtenção das *wavelets*-filhotes, $\psi_{j,k}(x)$, através de suas dilações ou compansões binárias (de 2^j) e deslocamentos diádicos (de $k/2^j$) então essa família de *wavelets* é conhecida como *wavelets* diádicas e é definida pela seguinte expressão

$$\psi_{j,k}(x) = 2^{j/2}\,\psi(2^j x - k), \qquad j,k \in Z \tag{2.16}$$

Escolhendo-se uma *wavelet*-mãe de norma unitária, isto é , $\|\psi\| = 1$, é fácil verificar que as *wavelets*-filhotes, $\psi_{j,k}(x)$, também possuem normas unitárias, isto é, $\|\psi_{j,k}\| = \|\psi\| = 1, j,k \in Z$, onde $\|f\| = \left[\int_{-\infty}^{\infty} |f(x)|^2\,dx\right]^{1/2}$.

Como já foi enfatizado, é possível reconstruir uma função de energia finita a partir dos seus coeficientes da TWC. No caso da TWD, entretanto, é preciso tomar cuidado ao se escolher a *wavelet* -mãe e a densidade da grade, de modo que a reconstrução não se torne instável [5].

A reconstrução do sinal $f(x)$, a partir de seus coeficientes *wavelet*, é dada pela seguinte expressão,

$$f(x) = c\sum_{j=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} C_{j,k}\,\psi_{j,k}(x), \tag{2.17}$$

onde os $C_{j,k}$ são os coeficientes da TWD e c é uma constante.

Casos Particulares

a) Grade Esparsa, por exemplo, $a_0 = 2$,

A representação *wavelet* se torna menos redundante. Como conseqüência, a expressão de reconstrução pode não ser mais válida, ou seja, a *wavelet* e os coeficientes correspondentes podem não ser mais adequados para a reconstrução do sinal. Assim, para se garantir a reconstrução nesse caso é preciso fazer uma série de restrições relativas a *wavelet* mãe $\psi(x)$.

b) Grade Densa, por exemplo, $a_0 = 1$,

As *wavelets* são redundantes e a reconstrução se dá com maior fidelidade. Além disso, não há muitas restrições acerca da *wavelet* $\psi(x)$ a serem utilizadas. A representação redundante, entretanto não é eficiente.

c) Ideal,

Corrresponde a grades densas apenas o suficiente para que a reconstrução seja realizada com um mínimo de perda e um máximo de eficiência. É usual buscar uma base ortonormal para se representar um sinal de forma eficiente.

Definição 2.2: Uma *wavelet* é ortonormal, se a sua família $\{\psi_{j,k}\}$ é definida na Eq. (2.16) e forma uma base ortonormal em $L^2(R)$, isto é, se [1, 4]

$$\left\langle \psi_{j,k}, \psi_{l,m} \right\rangle = \delta_{j,k}\delta_{k,m} \qquad\qquad j,k,l,m \in Z,$$

e se toda função $f(x) \in L^2(R)$ pode ser representada por

$$f(x) = \sum_{j=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} C_{j,k}\psi_{j,k}(x), \tag{2.18}$$

onde, tendo-se em vista que as funções $\psi_{j,k}$ são ortonormais, os coeficientes $C_{j,k}$ podem ser facilmente calculados através de,

$$C_{j,k} = \left\langle f, \psi_{j,k} \right\rangle.$$

A série (2.18) converge em $L^2(R)$, isto é

$$\lim_{M_1,N_1,M_2,N_2\to\infty} \left\| f - \sum_{j=-M_2}^{N_2}\sum_{j=-M_1}^{N_2} C_{j,k}\psi_{j,k} \right\| = 0.$$

III.2. "*FRAMES*" (QUADROS)

É importante, para se obter uma reconstrução numericamente estável de f a partir dos coeficientes $C_{j,k}$, que os vetores utilizados para a expansão sejam constituídos por "*frames*" [9]. Portanto nesta seção, apresenta-se um breve estudo dos "*frames*". Detalhes e considerações mais rigorosos sobre os "*frames*" podem ser consultados em [11, 12].

Um *frame* é um conjunto de elementos funcionais tais que qualquer função não-nula deve ter uma projeção nula pelo menos em um desses elementos. Além disso, qualquer função finita pode ser representada através de uma soma de suas projeções.

Definição 2.3: Seja $\{g_k, k \in K\}$ um conjunto de funções $L^2(R)$. Esse conjunto é um *frame* se existirem números $0 \le A \le B < \infty$, tais que

$$A\|f\|^2 \le \sum_{k\in K} |\langle g_k, f\rangle|^2 \le B\|f\|^2, \qquad \forall f \in L^2(R), \tag{2.19}$$

onde os números A e B são conhecidos como limitantes do *frame* [9].

Os elementos de um *frame* podem ser linearmente dependentes, ao contrário do que acontece com os elementos de uma base. Um *frame* é necessariamente um conjunto completo em $L^2(R)$. Entretanto pelo fato de o *frame* ser geralmente redundante, ele não necessariamente constitui uma base para $L^2(R)$. A dependência entre os elementos do *frame* cria uma redundância, o que sugere que a representação pode não ser única.

III.2.1 TIPOS DE *FRAMES*

- ***frames* Exatos:** *frames* não - redundantes. Um *frame* é exato se e somente se, ele é uma base para $L^2(R)$ (não necessariamente uma base ortonormal).
- ***frames* Estreitos**: Se os seus limitantes, A e B são iguais. Nesse caso se $\|g_k\|=1$, $A = B$ fornece a taxa de redundância ou taxa de sobre-amostragem. Se essa taxa é igual a 1, obtem-se o caso de amostragem crítico, então o *frame* é uma base ortonormal. Essas observações levarão ao enunciado que se apresenta a seguir [3].

Lema 2.1: Se o *frame* é estreito, $A = B = 1$ e todos os componentes do *frame*, $\{g_k, k \in K\}$, têm norma unitária, $\|g_k\| = 1$, então o *frame* é uma base ortonormal.

A qualquer *frame* $\{g_k, k \in K\}$, pode-se associar um "operador *frame*" T: $L^2(R) \rightarrow l^2(Z^2)$ definido por

$$Tf = \sum_{k \in K} \langle f, g_k \rangle g_k \,, \tag{2.20}$$

ou seja, a transformação Tf consiste na expansão em série de f, onde o *frame* $\{g_k\}$ é utilizado como base e os coeficientes são dados por $\langle f, g_k \rangle$.

Demonstra-se em [11] que o operador T é inversível. A operação inversa representa-se aqui por T^{-1}.

Lema 2.2: Seja $\{\gamma_k, k \in K\}$ um conjunto de funções obtidas a partir do *frame* $\{g_k\}$ através da seguinte operação

$$\gamma^k = T^{-1} g_k \,, \tag{2.21}$$

O conjunto $\{\gamma_k\}$ é um *frame* definido como *frame* dual de $\{g_k\}$ [11].

Quando um *frame* é exato, os conjuntos $\{g_k\}$ e $\{\gamma_k\}$ são biortogonais, ou seja

$$\langle g_k, \gamma_k \rangle = \delta(k - l) \tag{2.22}$$

Teorema 2.1: Para qualquer *frame* $\{g_k\}$ e o seu dual $\{\gamma_k\}$, pode-se expressar qualquer função $f(x) \in L^2(R)$ através de qualquer uma das seguintes formas,

$$f = \sum_{k \in K} \langle f, \gamma_k \rangle g_k \,, \tag{2.23}$$

e

$$f = \sum_{k \in K} \langle f, g_k \rangle \gamma_k \tag{2.24}$$

Essas equações mostram que [6]:

1) Pode-se expressar qualquer função de L^2(R) como uma combinação linear dos elementos de um *frame*.
2) O conjunto de coeficientes das Eqs.(2.23) e (2.24) é quadraticamente somável, o que é conseqüência do fato de que $\{g_k\}$ e $\{\gamma_k\}$ formam um par de *frames* duais.
3) Dado um *frame* $\{g_k\}$, para se obter f(x) precisa-se apenas calcular o seu dual $\{\gamma_k\}$, o que é feito através da expressão (2.22).

4) A expressão (2.23) é muito semelhante a uma expansão em base ortonormal, mas suas propriedades são diferentes. Em particular, os seus coeficientes na Eq.(2.23) são obtidos a partir de produtos internos da função com os elementos do *frame* dual, ao invés de produtos internos da função com os elementos do *frame*, como acontece na expansão ortogonal.

Se uma *wavelet* $\psi(x)$ gera um *frame*, ou seja, se $\psi(x)$ satisfaz a seguinte relação

$$A\|f\|^2 \le \sum_{j,k\in Z} \left|\langle f, \psi_{j,k}\rangle\right|^2 \le B\|f\|^2, \tag{2.25}$$

então, uma função $f(x) \in L^2(R)$ pode ser recuperada a partir dos coeficientes da sua transformada *wavelet* [4].

Seja $\{\psi_{j,k}\}$ um *frame*. Substituindo $\{g_k\} = \{\psi_{j,k}\}$ na Eq.(2.20), obtém-se a seguinte expressão,

$$Tf = \sum_{j,k\in Z} \langle f, \psi_{j,k}\rangle \psi_{j,k}, \qquad f \in L^2(R). \tag{2.26}$$

De acordo com o Teorema 2.1, a função $f(x)$ pode ser expressa na forma

$$f = T^{-1}T\, f = \sum_{j,k\in Z} \langle f, \psi_{j,k}\rangle\, T^{-1}\psi_{j,k}, \tag{2.27}$$

Definindo-se $\tilde{\psi}_{j,k}$ como o *frame* dual de $\psi_{j,k}$

$$\tilde{\psi}_{j,k} = T^{-1}\psi_{j,k} \qquad j,k \in Z, \tag{2.28}$$

a fórmula de reconstrução da função $f(x)$ pode ser reescrita na forma

$$f = \sum_{j,k\in Z} \langle f, \psi_{j,k}\rangle \tilde{\psi}_{j,k}, \tag{2.29}$$

ou

$$f = \sum_{j,k\in Z} \langle f, \tilde{\psi}_{j,k}\rangle \psi_{j,k}. \tag{2.30}$$

No caso de *frames* estreitos, o *frame* dual é composto de versões escalonadas e deslocadas da *wavelet*-mãe, multiplicadas por uma constante isto é,

$$\tilde{\psi}_{j,k} = A^{-1}\psi_{j,k}, \tag{2.31}$$

e, para esse caso, pode-se reescrever a Eq.(2.29) na forma

$$f = A^{-1} \sum_{j,k\in Z} \langle f, \psi_{j,k}\rangle \psi_{j,k} \tag{2.32}$$

Os *frames* que obedecem às condições impostas pelo *Lema* 2.1 formam bases ortonormais. Para esses, a expressão (2.32) pode ser simplificada como

$$f = \sum_{j,k \in Z} \left\langle f, \psi_{j,k} \right\rangle \psi_{j,k} \tag{2.33}$$

***Frames* Redundantes versus Ortonormais** [4]

- Os *frames* redundantes permitem que uma maior classe de funções possam ser usadas como *wavelet*-mãe.
- Os vetores de um *frame* redundante podem, "casar" melhor com um sinal, e, portanto, um número menor de coeficientes será necessário para representar o conteúdo de informação.
- Os *frames* ortonormais impõem restrições à *wavelet*-mãe, o que pode não ser desejável em algumas aplicações.
- Para representação, processamento e codificação de imagens, as transformadas *wavelets* ortonormais são mais adequadas, por serem mais facilmente implementáveis, dada sua baixa complexidade computacional.

2.2. MULTIRESOLUÇÃO

I. INTRODUÇÃO

A análise da informação contida em uma imagem diretamente a partir da intensidade, da escala de cinza dos *pixels*, é um processo muito dificultoso. A representação de multiresolução fornece uma base para a interpretação e análise da informação contida em um sinal. Essa informação é organizada em um conjunto de detalhes com diferentes resoluções. Para um sinal $f(x) \in L^2(R)$, definido no intervalo $0 < x < x_0$, a sua resolução nos fornece a quantidade de pontos para os quais esse sinal é conhecido. Quando se conhece um sinal com resolução completa, ou o que é o mesmo com um nível de resolução infinita, pode-se dizer que se conhece $f(x)$ para todos os pontos do intervalo. Da mesma forma, pode-se dizer que se conhece um sinal com resolução r_j quando se conhece r_j pontos do intervalo.

Dada uma seqüência crescente de resoluções $(r_j)_{j \in Z}$, os detalhes com resolução r_j de um sinal são definidos como sendo a diferença de informação entre a sua aproximação com resolução r_j e a sua aproximação com resolução r_{j-1}, mais baixa [12, 13].

II. NOTAÇÃO

A_{2^j} é o operador que gera uma aproximação com resolução 2^j de um sinal.

O sinal original $f(x)$ é mensurável e tem energia finita: $f(x) \in L^2(R)$

III. PROPRIEDADES DO OPERADOR A_{2^j}

1. A_{2^j} é um operador linear. Se $A_{2^j} f(x)$ é uma aproximação de uma função $f(x)$ com resolução 2^j, então $A_{2^j} f(x)$ não é modificada quando se repete a operação no nível de resolução 2^j. Isso significa que $A_{2^j} \circ A_{2^j} = A_{2^j}$. O operador A_{2^j} é então, um operador de projeção no espaço vetorial $V_{2^j} \subset L^2(R)$. O espaço vetorial V_{2^j} pode ser

interpretado como sendo o conjunto de todas as possíveis aproximações das funções $L^2(R)$ no nível de resolução 2^j.

2. Dentre todas as aproximações da função $f(x)$, no nível de resolução 2^j, $A_{2^j} f(x)$ é a função que mais se aproxima de $f(x)$, ou seja,

$$\forall g(x) \in V_{2^j}, \left\| g(x) - f(x) \right\| \geq \left\| A_{2^j} f(x) - f(x) \right\| \tag{2.34}$$

Portanto, o operador A_{2^j} calcula a projeção ortogonal de uma função no espaço vetorial V_{2^j}

3. A aproximação de um sinal no nível de resolução 2^{j+1} contém toda a informação necessária para se calcular a sua aproximação no nível de resolução 2^j. Este é o princípio da causalidade. Como A_{2^j} é um operador de projeção em V_{2^j},essa propriedade é equivalente a

$$\forall j \in Z, \quad V_{2^j} \subset V_{2^{j+1}} \tag{2.35}$$

4. A operação de aproximação é similar para todos os níveis de resolução. Os subespaços das funções podem ser obtidos a partir dos outros subespaços, através da dilatação ou compressão das aproximações das funções na razão dos valores das suas resoluções. Isso significa que,

$$\forall j \in Z, \quad A_{2^j} f(x) \in V_{2^j} \Rightarrow A_{2^j} f(2x) \in V_{2^{j+1}} \tag{2.36}$$

5. A aproximação $A_{2^j} f(x)$ pode ser caracterizada por 2^j amostras por unidade de comprimento. Quando $f(x)$ é transladada por um comprimento proporcional a 2^{-j}, $A_{2^j} f(x)$ é transladada pela mesma quantidade e é caracterizada pelas mesmas amostras que foram deslocadas. Como conseqüência da propriedade 3, é suficiente expressar a propriedade 5 para o nível de resolução $j = 0$.

6. Quando se calcula uma aproximação de $f(x)$ no nível de resolução 2^j, alguma informação sobre $f(x)$ é perdida. À medida que a resolução aumenta em direção a $+\infty$, o sinal aproximação converge para o sinal original. Da mesma forma, à medida que a resolução diminui quando se aproxima do zero, o sinal aproximado passa a conter cada vez menos informação e, portanto, converge para zero. Como a aproximação $A_{2^j} f(x)$ é

igual à projeção ortogonal de $f(x)$ no espaço vetorial V_{2^j}, esta propriedade pode ser escrita da seguinte forma

$$\lim_{j\to+\infty} V_{2^j} = \bigcup_{j=-\infty}^{+\infty} V_{2^j} = L^2(R), \tag{2.37}$$

e

$$\lim_{j\to-\infty} V_{2^j} = \bigcap_{-\infty}^{+\infty} V_{2^j} = \{0\}. \tag{2.38}$$

IV. APROXIMAÇÃO POR MULTIRESOLUÇÃO EM L^2 (R)

***Definição 2.4*:** Denominamos de aproximação por multiresolução de $L^2(R)$, qualquer conjunto de espaços vetoriais $\{V_{2^j}\}_{j\in Z}$ que satisfazem as propriedades 1-6. O conjunto de operadores A_{2^j} associados a esse conjunto de espaços vetoriais gera as aproximações de qualquer função $f(x) \in L^2(R)$ em todos os níveis de resolução 2^j, para $j \in Z$ [12, 13].

Teorema 2.2: Seja $(V_{2^j})_{j\in Z}$ uma aproximação por multiresolução em $L^2(R)$. Existe uma única função $\phi(x) \in L^2(R)$, denominada função escala, tal que quando se define a dilatação de $\phi(x)$ por 2^j, como sendo

$$\phi_{2^j}(x) = 2^j \phi(2^j x) \quad , \tag{2.39}$$

para $j \in Z$, então o conjunto $\left(2^{-j/2}\, \phi_{2^j}(x - 2^{-j} n)\right)_{n\in Z}$ é uma base ortonormal para V_{2^j} [12].

Esse conjunto também pode ser expresso de modo equivalente neste trabalho como,

$$\left(2^{-j/2}\, \phi_{2^j}(x - 2^{-j} n)\right) = \left(2^{-j/2}\, 2^j\, \phi\left(2^j (x - 2^{-j} n)\right)\right) = \left(2^{j/2}\, \phi\, (2^j x - n)\right) = \phi_{j,n}(x) \tag{2.40}$$

A partir desse Teorema mostra-se que [13]:

- Dilatando-se ou comprimindo-se uma função $\phi(x)$ por um fator 2^j e deslocando-se a função resultante em intervalos proporcionais a 2^{-j}, pode-se construir uma base ortonormal em qualquer espaço vetorial V_{2^j}.

- As funções $\phi_{2^j}(x)$ são normalizadas com relação a $L^1(R)$.
- O coeficiente $2^{-j/2}$ tem o objetivo de normalizar as funções em $L^2(R)$.
- Para uma dada aproximação por multiresolução $(V_{2^j})_{j\in Z}$ existe uma única função escala $\phi(x)$ que satisfaz a Eq.(2.39). Entretanto para diferentes aproximações existem diferentes funções de escalonamento.
- A aproximação por multiresolução $(V_{2^j})_{j\in Z}$ é completamente caracterizada pela função escala $\phi(x)$. Essa pode ser definida como uma função $\phi(x) \in L^2(R)$, tal que, para qualquer $j \in Z$, $\left(2^{-j/2}\,\phi_{2^j}(x-2^{-j}n)\right)_{n\in Z}$ é uma família ortonormal.
- Se V_{2^j} é o espaço vetorial gerado por essa família, $(V_{2^j})_{j\in Z}$ é uma aproximação por multiresolução em $L^2(R)$.
- A função $\phi(x)$ deve ser continuamente diferenciável. O decaimento assintótico de $\phi(x)$ e de sua derivada $\phi'(x)$ no infinito deve satisfazer às seguintes relações:

$$\phi(x) = O(x^{-2}) \quad e \quad |\phi'(x)| = O(x^{-2})$$

- A projeção ortogonal de uma função $f(x)$ no espaço V_{2^j} pode ser calculada decompondo-se essa função na base ortogonal definida pelo Teorema 2.2, ou seja,

$$\forall f(x) \in L^2(R),\ A_{2^j} f(x) = 2^{-j} \sum_{n=-\infty}^{\infty} \left\langle f(u), \phi_{2^j}(u-2^{-j}n)\right\rangle \phi_{2^j}(x-2^{-j}n) \qquad (2.41)$$

- A aproximação do sinal $f(x)$ no nível de resolução 2^j, $A_{2^j} f(x)$, é então, caracterizada pelo seguinte conjunto de produtos internos,

$$A_{2^j}^d f = \left(\left\langle f(u), \phi_{2^j}(u-2^{-j}n)\right\rangle\right)_{n\in Z} \qquad (2.42)$$

onde $A_{2^j}^d f$ é denominada, *aproximação discreta* do sinal $f(x)$ no nível de resolução 2^j.

- Cada produto interno pode ser interpretado também, como uma convolução avaliado no ponto $2^{-j}n$, isto é,

$$\left\langle f(u), \phi_{2^j}(u-2^{-j}n)\right\rangle = \int_{-\infty}^{\infty} f(u)\phi_{2^j}(u-2^{-j}n)du = \left(f(u) * \phi_{2^j}(-u)\right)(2^{-j}n) \qquad (2.43a)$$

então, rescreve-se $A_{2^j}^d f$, como sendo,

$$A_{2^j}^d f = \left(\left(f(u) * \phi_{2^j}(-u)\right)\left(2^{-j} n\right)\right)_{n \in Z} \tag{2.43b}$$

Desde que $\phi(x)$ é um filtro passa-baixas, o sinal discreto obtido é o resultado de uma filtragem passa-baixas sobre $f(x)$, seguido de uma amostragem uniforme à taxa de 2^j.

- A função de escala $\phi(x)$, constitui uma forma muito particular de filtro passa – baixas devido a que a família de funções $\left(2^{-j/2}\,\phi_{2^j}(x - 2^{-j} n)\right)_{n \in Z}$, é uma família ortonormal.

A aproximação discreta do sinal $f(x)$ no nível de resolução 2^j pode ser calculado com o algoritmo piramidal, o qual é mostrado na próxima seção.

V. Implementação de uma Transformada de Multiresolução

Na prática os sinais medidos possuem uma resolução máxima finita. Por uma questão de normalização, supõe-se [13]:

- Resolução máxima igual a 1.
- $A_{2^j}^d f$ é a aproximação discreta nesta resolução máxima .

O princípio da causalidade diz, que a partir de $A_{2^j}^d f$ pode-se calcular todas as aproximações discretas $A_{2^j}^d f$ para $j < 0$.

Nesta seção, descrevem-se algoritmos interativos simples para se calcular essas aproximações discretas.

Seja $\left(V_{2^j}\right)_{j \in Z}$ uma aproximação por multiresolução e $\phi(x)$ a função de escalonamento correspondente. De acordo com o Teorema 2.2, a família de funções $\left(\sqrt{2^{-j-1}}\,\phi_{2^{j+1}}(x - 2^{-j-1} n)\right)_{k \in Z}$ é uma base ortonormal para $V_{2^{j+1}}$. Sabe-se que para qualquer $n \in Z$, a função $\phi_{2^j}(x - 2^{-j} n)$ pertence ao espaço V_{2^j} que por sua vez, está contido em $V_{2^{j+1}}$. Então pode-se expandir a função na base ortonormal de $V_{2^{j+1}}$:

A partir da Eq.(2.40) obtem-se,

$$\left\langle f(x), \phi_{j,n}(x) \right\rangle = \sum_{k=-\infty}^{\infty} \left\langle \phi_{-1,0}(u), \phi_{0,k-2n}(u) \right\rangle \cdot \left\langle f(u), \phi_{j+1,k}(x) \right\rangle \tag{2.44}$$

Sob o ponto de vista do processamento digital de sinais, a Eq.(2.44) pode ser interpretada como sendo uma filtragem e pode-se definir H como sendo o filtro de resposta impulsiva, dada por,

$$h(n) = \left\langle \phi_{-1,0}(u), \phi_{0,n}(u) \right\rangle \qquad \forall n \in Z \tag{2.45}$$

desde que $\widetilde{H}$ é o filtro espelhado *("mirror filter")* com resposta impulsiva $\widetilde{h}(n) = h(-n)$. Substituindo-se a Eq.(2.44) na Eq.(2.45), obtem-se,

$$\left\langle f(x), \phi_{j,n}(x) \right\rangle = \sum_{k=-\infty}^{\infty} h(k-2n) \cdot \left\langle f(x), \phi_{j+1,k}(x) \right\rangle = \sum_{k=-\infty}^{\infty} \widetilde{h}(2n-k) \cdot \left\langle f(x), \phi_{j+1,k}(x) \right\rangle \tag{2.46}$$

A expressão (2.46) mostra que [13]:

- $A^d_{2^j} f$ pode ser calculada convoluindo-se $A^d_{2^{j+1}} f$ com h (-n) e tomando-se uma amostra sim e outra não do sinal resultante (subamostragem por um fator 2)
- Todas as aproximações discretas $A^d_{2^j} f$, para $j < 0$, podem ser calculadas a partir de $A^d_{2^j} f$ através da aplicação recursiva dessa operação, que é denominada transformada piramidal [14].

O algoritmo é mostrado na Fig.2.3. Observe que o símbolo **h (-n)** indica a convolução do sinal com o filtro h (-n) e o $2\downarrow$ indica o subamostragem do resultado.

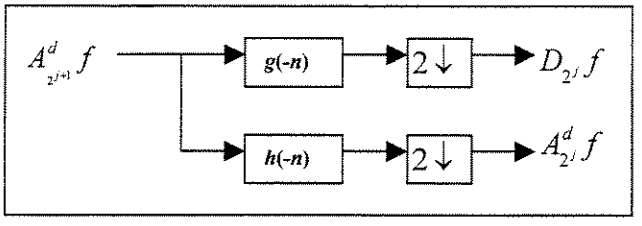


Fig. 2.3: Diagrama em blocos do esquema de decomposição de uma aproximação discreta $A^d_{2^{j+1}} f$.

Na prática, os sinais medidos terão um número finito de amostras. Suponha-se agora que a aproximação discreta do sinal $f(x)$ no maior nível de resolução é completamente caracterizada por N coeficientes, ou seja, $A^d_1 f = (\alpha_n)_{1 \le n \le N}$. Cada aproximação discreta $A^d_{2^j} f \quad (j < 0)$ tem, portanto, $2^j N$ amostras. De maneira a evitar problemas de borda ao se calcular as aproximações discretas $A^d_{2^j} f$, suponha-se também que o sinal original $A^d_{2^j} f$ é simétrica em relação a n =0 e N - 1, ou seja:

$$\alpha_n = \begin{cases} \alpha_{-n} & se\ -N-1 < n < 0 \\ \alpha_{2N-n} & se\ \ 0 < n < N-1 \end{cases}$$

Se a resposta impulsiva do filtro H for par, isto é, $h(-n) = h(n)$, então cada aproximação discreta $A^d_{2^j} f$ também será simétrica com relação a $n = 0$ e $n = 2^{-j} N$ [15].

V.1. INTERPRETAÇÃO DA FILTRAGEM DIGITAL

Utilizando-se a definição dada na Eq.(2.45), pode-se escrever,

$$\phi_{j,n}(x) = \sum_{k=-\infty}^{\infty} h(k-2n)\phi_{j+1,k}(x) \tag{2.47}$$

Fazendo-se $j = 0$ e $n = 0$, obtem-se,

$$\phi(x) = \sum_{k=-\infty}^{\infty} h(k) 2^{1/2} \phi(2x-k). \tag{2.48}$$

Essa é a expressão fundamental da análise de multiresolução e levará ao algoritmo de decomposição da transformada *wavelet*. A seqüência $h(k)$ pode ser interpretada como sendo um filtro discreto [12].

Lema 2.3:Se a seqüência $\{h(k)\}$ é não nula apenas para $0 \le k \le N-1$, ou seja, se $h(k)$ é um filtro FIR, então $\phi(x)$ é não nula apenas para $0 \le k \le N-1$ [11].

Aplicando-se a transformada de Fourier definida na Eq.(2.48), obtem-se

$$\Phi(\omega) = \sum_{k=-\infty}^{\infty} h(k) \frac{\Phi(0.5\omega)}{2^{1/2}} e^{-j0.5\omega k} = \frac{\Phi(0.5)}{2^{1/2}} H(0.5\omega). \tag{2.49}$$

Uma conseqüência imediata da Eq.(2.49) é obtida para $\omega = 0$,

$$H(0) = \sum_{k=-\infty}^{\infty} h(k) = \sqrt{2} \tag{2.50}$$

Aplicando-se a Eq.(2.46) recursivamente,

$$\Phi(\omega) = \Phi(0) \prod_{m=1}^{\infty} \frac{H(\omega 2^{-m})}{2^{1/2}} \tag{2.51}$$

Quando a Eq.(2.51) converge, pode-se obter um algoritmo recursivo para a construção de $\phi(x)$.

V.2. ALGORITMO RECURSIVO PARA A GERAÇÃO DE $\phi(x)$ A PARTIR DOS COEFICIENTES DO FILTRO [16, 17, 18].

1. Seja $\phi(x)$, a função escala inicial, conhecida como função escala de Haar

$$\phi(x) = \begin{cases} 1 & 0 \le t \le 1 \\ 0 & c.c. \end{cases}$$

2. Para $j \ge 0$, tem-se

$$\phi_j(x) = \sum_{k=-\infty}^{\infty} h(k) 2^{1/2} \phi_{j+1}(2x-k). \tag{2.52}$$

Essa expressão permite obter a função escala da próxima iteração.

3. O processo iterativo é realizado, até, que uma divergência seja encontrada ou a desejada convergência seja obtida. Se existe convergência, a função escala é dada por

$$\phi(x) = \lim_{j \to \infty} \phi_j(x). \tag{2.53}$$

A prova da convergência desse algoritmo se encontra em [16].

Teorema 2.3 Seja $\phi(x)$ uma função escala e H o filtro discreto com resposta impulsiva,

$$h(n) = \left\langle \phi_{-1,0}(u), \phi_{0,n}(u) \right\rangle$$

Seja $H(\omega)$ a série de Fourier de $h(n)$, definida por

$$H(\omega) = \sum_{n=-\infty}^{\infty} h(n) e^{-in\omega}, \tag{2.54}$$

onde $H(\omega)$ satisfaz as seguintes propriedades:

$$H(0) = \sqrt{2} \quad \text{e} \quad h(n) = O(x^{-2}) \text{ no infinito} \tag{2.55}$$

$$|H(\omega)|^2 + |H(\omega + \pi)|^2 = 2. \tag{2.56}$$

Essas propriedades indicam que $H(\omega)$ é um filtro passa-baixas. Se $H(\omega)$ satisfaz as igualdades dadas nas Eqs.(2.55) e (2.56) e, ainda apresenta a seguinte característica:

$$|H(\omega)| \ne 0 \qquad para\ \omega \in \left[0, \pi/2\right] \tag{2.57}$$

então, a função definida por:

$$\Phi(\omega) = \Phi(0) \prod_{p=1}^{\infty} \frac{H(2^{-p}\omega)}{2^{1/2}} \tag{2.58}$$

é a transformada de Fourier da função escala [19]. As provas desse teorema podem ser encontradas em [12].

Os filtros que satisfazem a propriedade dada na Eq.(2.56) são chamados de filtros conjugados. A partir de um filtro conjugado $H(\omega)$ pode-se calcular a transformada de Fourier da função escala correspondente através da Eq.(2.58). É possível, então escolher $H(\omega)$ de maneira a se obter uma função escala $\phi(x)$ com boas propriedades de localização, tanto no domínio da freqüência quanto no domínio espacial [4, 13].

V.3. O SINAL DE DETALHAMENTO

Nesta seção mostramos que a representação pode ser calculada pela decomposição do sinal, usando-se uma base *wavelet* ortonormal.

Sinal de Detalhamento no nível de resolução 2^j [13]: Sinal formado a partir da diferença de informação entre os níveis de resolução 2^j e 2^{j+1}

Como já é conhecido, as aproximações de um sinal nos níveis de resolução 2^{j+1} e 2^j, são respectivamente iguais às suas projeções ortogonais em $V_{2^{j+1}}$ e V_{2^j}. Como conseqüência, o sinal de detalhamento no nível de resolução 2^j é obtido através da projeção ortogonal do sinal original no complemento ortogonal de V_{2^j} em $V_{2^{j+1}}$.

Seja W_{2^j} o complemento ortogonal de V_{2^j}, ou seja

$$W_{2^j} \text{ é ortogonal a } V_{2^j},$$

$$V_{2^j} \oplus W_{2^j} = V_{2^{j+1}}.$$

Para se calcular a projeção ortogonal de uma função $f(x)$ em W_{2^j}, é necessário encontrar uma base ortonormal para gerar o espaço W_{2^j}. Assim:

- Deve-se procurar uma função, $\psi(x)$ que tenha para W_{2^j} o mesmo papel que $\phi(x)$ tem para V_{2^j}.
- Deseja-se que a família definida por, $\left\{\psi_{j,k}(x) = 2^{j/2}\,\psi(2^j x - k),\ k \in Z\right\}$, seja uma base ortonormal para W_{2^j}, o que significa que $\left\{\psi_{j,k}(x),\ j, k \in Z\right\}$, é uma base ortonormal para $L^2(R)$.
- Demonstra-se que a base procurada pode ser formada a partir de uma família de *wavelets*, ou seja, que as *wavelets* podem ser utilizadas como funções geradoras dos subespaços W_{2^j}.

- Assume-se a existência dessa função e verificam-se as conseqüências dessa suposição.

Esse procedimento fornece um algoritmo para a construção de $\psi(x)$.

Para qualquer $n \in Z$, a função $\psi_{j,n}(x)$ pertence a $W_{2^j} \subset V_{2^{j+1}}$. Portanto, essa função pode ser expandida na base ortonormal de $V_{2^{j+1}}$

$$\psi_{j,n}(x) = \sum_{k=-\infty}^{\infty} \left\langle \psi_{-1,0}(u), \phi_{0,k-2n}(u) \right\rangle \phi_{j+1,k}(x) . \tag{2.59}$$

Pode-se interpretar a Eq.(2.59) como sendo uma filtragem, onde G é definido como um filtro discreto de resposta impulsiva dada por

$$g(n) = \left\langle \psi_{-1,0}(u), \phi_{0,n}(u) \right\rangle . \tag{2.60}$$

Substituindo-se a Eq.(2.60) na Eq.(2.59), tem-se

$$\psi_{j,n}(x) = \sum_{k=-\infty}^{\infty} g(k-2n)\phi_{j+1,k}(x), \tag{2.61}$$

Fazendo-se $j = 0$ e $n = 0$ em (2.61), obtem-se

$$\psi(x) = \sum_{k=-\infty}^{\infty} g(k) 2^{1/2} \phi(2x-k), \tag{2.62}$$

onde $\{g(k)\}$ é uma seqüência quadraticamente somável. Tomando-se a transformada de Fourier de ambos lados da Eq.(2.62), obtem-se

$$\Psi(\omega) = \sum_{k=-\infty}^{\infty} g(k) \frac{\Phi(0.5\omega)}{2^{1/2}} e^{-j0.5\omega n} = \frac{\Phi(0.5\omega)}{2^{1/2}} G(0.5\omega), \tag{2.63}$$

Lema 2.3: A família $\{\psi_{j,n}\}_{n \in Z}$ é uma base ortonormal de W_{2^j} se e somente se,

$$|G(\omega)|^2 + |G(\omega+\pi)|^2 = 2, \tag{2.64}$$

e

$$H(\omega)G^*(\omega) + H(\omega+\pi)G^*(\omega+\pi) = 0, \tag{2.65}$$

A prova deste Lema encontra-se [12].

Para mostrar que $\{\psi_{j,k}(x),\ k \in Z\}$ é uma base ortonormal, resta apenas verificar se $\psi(x)$ satisfaz a condição de admissibilidade, ou seja, se $\Psi(0) = 0$. A Eq.(2.50) mostra que $H(0) = \sqrt{2}$. Substituindo-se a Eq.(2.50) na Eq.(2.56), obtem-se

$$|H(0)|^2 + |H(\pi)|^2 = 2 + |H(\pi)|^2 = 2$$
$$|H(\pi)|^2 = 0 \tag{2.66}$$

Substituindo-se a Eq.(2.66) na Eq.(2.65), obtem-se

$$G(0) = 0,$$

e

$$\Psi(0) = \frac{\Phi(0)}{\sqrt{2}} G(0) = 0.$$

Pode-se concluir que $\Psi(\omega)$ satisfaz a condição de admissibilidade e, portanto, $\{\psi_{j,k}(x),\ k \in Z\}$ é uma base *wavelet*.

Uma vez fixado $H(\omega)$, $G(x)$ não é mais arbitrário. Para provar essa afirmação define-se

$$H_i = H(\omega) \quad e \quad H_j = H(\omega + \pi), \tag{2.67}$$

$$G_i = G(\omega) \quad e \quad G_j = G(\omega + \pi), \tag{2.68}$$

Substituindo-se as Eqs.(2.67) e (2.68) na Eq.(2.66), obtem-se

$$\frac{G_i}{G_j} = -\left(\frac{H_j}{H_i}\right)^*,$$

Dividindo-se ambos os lados dessa expressão por $\left[1 + \left|\frac{G_i}{G_j}\right|^2\right]^{1/2} = \left[1 + \left|\frac{H_j}{H_i}\right|^2\right]^{1/2}$, tem-se

$$\frac{\frac{G_i}{G_j}}{\left[1 + \left|\frac{G_i}{G_j}\right|^2\right]^{1/2}} = \frac{-\left(\frac{H_j}{H_i}\right)^*}{\left[1 + \left|\frac{H_j}{H_i}\right|^2\right]^{1/2}}, \tag{2.69}$$

Multiplicando-se o numerador e o denominador do lado esquerdo da Eq.(2.69) por $|G_j|$, os correspondentes do lado direito por H_i e substituindo-se as Eqs.(2.65) e (2.56) no resultado, obtem-se

$$G_i = -\frac{|H_i|}{|G_j|}\frac{G_j}{H_i^*}H_j^* = -A(\omega)H_j^*, \tag{2.70}$$

onde $A(\omega)$ é uma função passa- tudo, pois $|A(\omega)| = 1, \quad \forall\omega$.

Substituindo-se a Eq.(2.70) na Eq.(2.65), constata-se que a condição,

$$A(\omega) + A(\omega + \pi) = 0, \tag{2.71}$$

é necessária e suficiente para que a Eq.(2.65) seja válida.

A escolha de $A(\omega) = e^{-j\omega}$ é muito freqüente. Substituindo-se esse valor de $A(\omega)$ na Eq.(2.70) obtem-se

$$G(\omega) = e^{-j\omega}H^*(\omega + \pi), \tag{2.71}$$

o que corresponde no domínio temporal a,

$$g(k) = (-1)^k h^*(1-k).$$

A Eq.(2.71) fornece uma relação entre os filtros H e G utilizados no processo de decomposição e sínteses da TW. Qualquer par de filtros que obedeça às condições apresentadas nesta seção pode ser utilizado. Entretanto, do ponto de vista de processamento de sinais, um filtro passa-baixas é uma escolha interessante para H. Nessa escolha, G é automaticamente um filtro passa-altas [20]. No processamento de sinais G e H são chamados de filtros espelhados em quadratura (*quadrature mirror filters*) [18, 21].

Com base nos resultados acima, pode-se então enunciar o próximo teorema, o qual estabelece uma relação entre a *wavelet* e a função de escalonamento.

Teorema 2.4: Sejam $\left(V_{2^j}\right)_{j\in Z}$ a seqüência geradora do conjunto de espaços vetoriais de uma análise de multiresolução, $\phi(x)$ a função escala e H o filtro conjugado correspondente. Seja $\psi(x)$ a função cuja transformada de Fourier $\Psi(\omega)$ é dada por

$$\Psi(\omega) = G\left(\frac{\omega}{2}\right)\Phi\left(\frac{\omega}{2}\right)2^{-1/2}, \tag{2.72}$$

com $G(\omega) = e^{-i\omega} H^*(\omega + \pi)$ e $\psi_{2^j}(x) = 2^j \psi(2^j x)$, então, tem-se que

$$\left(2^{-j/2} \psi_{2^j}(x - 2^{-j} n)\right)_{n \in Z},$$

é uma base ortonormal em W_{2^j} e $\psi(x)$ é uma *wavelet* ortogonal [13].

Neste trabalho, o conjunto anterior pode ser expresso de forma equivalente como,

$$\left(2^{-j/2} \psi_{2^j}(x - 2^{-j} n)\right) = \left(2^{-j/2} 2^j \psi(2^j (x - 2^{-j} n))\right) = \left(2^{j/2} \psi(2^j x - n)\right) = \psi_{j,n}(x).$$

A partir do *Teorema 2.4* mostra-se que:

- Uma base ortonormal em W_{2^j} pode ser calculada mediante o escalonamento da *wavelet* $\psi(x)$ por um coeficiente 2^j e transladando-se a função resultante, sobre uma grade cujo intervalo é proporcional a 2^{-j} [4].
- A partir de um par de filtros H e G, que satisfaçam as condições expressas nas Eqs.(2.55), (2.56) e (2.57), pode-se encontrar uma função escala e uma *wavelet*, utilizando-se as expressões (2.58) e (2.72), respectivamente.
- Dependendo da escolha de H, a função escala $\phi(x)$ e a *wavelet* $\psi(x)$ podem ter boa localização tanto no domínio espacial quanto no temporal [4].
- Seja $P_{W_{2^j}}$ o operador projeção ortogonal no espaço W_{2^j}. Como conseqüência do Teorema 2.4, pode-se escrever,

$$P_{W_{2^j}} f(x) = \sum_{n=-\infty}^{\infty} \langle f(u), \psi_{j,n}(u) \rangle \psi_{j,n}(x), \tag{2.73}$$

$P_{W_{2^j}} f(x)$, fornece o sinal de detalhamento de $f(x)$ com resolução 2^j e é caracterizado pelo conjunto de produtos internos,

$$D_{2^j} f = \left(\langle f(u), \psi_{j,n}(u) \rangle \right)_{n \in Z}, \tag{2.74}$$

$D_{2^j} f$, é o sinal discreto de detalhamento com resolução 2^j, que contém a diferença de informação entre $A^d_{2^{j+1}} f$ e $A^d_{2^j} f$.

V.4. IMPLEMENTAÇÃO DE UMA REPRESENTAÇÃO *WAVELET* ORTOGONAL

Nesta seção descreve-se o algoritmo piramidal utilizado para se calcular a representação *wavelet*. Mostra-se que $D_{2^j} f$ pode ser calculado através da convolução de $A^d_{2^{j+1}} f$ com o filtro G, cuja forma será caracterizada, pela seguinte expressão

$$\left\langle f(x), \psi_{j,n}(x) \right\rangle = \sum_{k=-\infty}^{\infty} g(k-2n) \left\langle f(x), \phi_{j+1,k}(x) \right\rangle, \tag{2.78}$$

A Eq.(2.78) mostra que se pode calcular o sinal de detalhamento $D_{2^j} f$ através da convolução de $A^d_{2^{j+1}} f$ com um filtro discreto de resposta impulsiva $g(-n)$, jogando-se fora uma amostra sim e outra não, conforme mostrado no ramo superior da Fig. 2.3.

O processo de decomposição é definido pelas expressões (2.46) e (2.78). A representação *wavelet* ortogonal do sinal discreto $A^d_{2^j} f$ pode ser calculada portanto, através de sucessivas decomposições de $A^d_{2^{j+1}} f$ em $A^d_{2^j} f$ e $D_{2^j} f$ para $J \le j \le -1$. Pode-se por indução, provar que para qualquer $J > 0$, o sinal original $A^d_{2^j} f$ pode ser representado por,

$$\left(A^d_{2^{-J}} f, \left(D_{2^j} f \right)_{-J \le j \le -1} \right). \tag{2.79}$$

Esse conjunto de coeficientes é denominado de representação *wavelet* ortogonal e é formado pela aproximação do sinal $A^d_{2^{-J}} f$ como uma resolução mais grosseira, e pelos sinais de detalhamento $D_{2^j} f$ nas resoluções 2^j, para $-J \le j \le -1$. O conjunto pode ser interpretado como sendo uma decomposição do sinal original em uma base *wavelet* ortonormal ou como a decomposição do sinal original em um conjunto de canais de freqüências independentes (*independent frequency channels*), [19]. Essa independência é devido à ortogonalidade das funções *wavelets*.

Se o sinal original tem N amostras, então os sinais discretos $D_{2^j} f$ e $A^d_{2^j} f$ têm $2^j N$ amostras cada e a representação *wavelet* dada pela Eq.(2.79) tem o mesmo número de amostras de $A^d_{2^j} f$. Isso se deve a ortogonalidade da representação.

Pode-se então, reescrever a Eq.(2.79) da seguinte forma

$$A^d_{2^1} f = A^d_{2^{-J}} f + \sum_{j=-J}^{-1} D_{2^j} f, \tag{2.80}$$

A expressão (2.80) representa o sinal decomposto em dois termos: o primeiro corresponde a aproximação da função $f(x)$ no nível de resolução $-J$ e o segundo aos sinais de detalhamento para todos os níveis de resolução $-J \le j \le -1$. Sob o ponto de vista de processamento de sinais, pode-se interpretar o primeiro termo da Eq.(2.80) como sendo a componente de baixas freqüências do sinal e o segundo como sendo a componente de altas freqüências.

V.5. RECONSTRUÇÃO DO SINAL DE UMA REPRESENTAÇÃO *WAVELET* ORTOGONAL

Foi visto que a representação *wavelet* é completa. Deseja-se mostrar agora que o sinal discreto original pode ser reconstruído com a transformação piramidal. Como W_{2^j} é o complemento ortogonal de V_{2^j} em $V_{2^{j+1}}$ então $\left\{\phi_{j,n}(x), \psi_{j,n}(x)\right\}_{n\in Z}$ é uma base ortonormal em $V_{2^{j+1}}$ [13]. Para qualquer, $n > 0$, pode-se decompor a função $\phi_{j+1,n}(x)$, em $V_{2^{j+1}}$,

$$\phi_{j+1,n}(x) = \sum_{k=-\infty}^{\infty} \left\langle \phi_{j+1,n}(u), \phi_{j,k}(u) \right\rangle \phi_{j,k}(x) + \sum_{k=-\infty}^{\infty} \left\langle \phi_{j+1,n}(u), \psi_{j,k}(u) \right\rangle \psi_{j,k}(x), \qquad (2.81)$$

Calculando-se o produto interno em ambos os lados da Eq.(2.81) com a função $f(x)$, tem-se

$$\begin{aligned} \left\langle f(x), \phi_{j+1,n}(x) \right\rangle = & \sum_{k=-\infty}^{\infty} \left\langle \phi_{j+1,n}(u), \phi_{j,k}(u) \right\rangle \left\langle f(x), \phi_{j,k}(x) \right\rangle \\ & + \sum_{k=-\infty}^{\infty} \left\langle \phi_{j+1,n}(u), \psi_{j,k}(u) \right\rangle \left\langle f(x), \psi_{j,k}(x) \right\rangle \end{aligned} \qquad (2.82)$$

Fazendo-se uma mudança de variáveis nos produtos internos na Eq.(2.82), tem-se

$$\left\langle \phi_{j+1,n}(u), \phi_{j,k}(u) \right\rangle = \left\langle \phi_{-1,0}(u), \phi_{0,n-2k}(u) \right\rangle = h(n-2k), \qquad (2.83)$$

$$\left\langle \phi_{j+1,n}(u), \psi_{j,k}(u) \right\rangle = \left\langle \psi_{-1,0}(u), \phi_{0,n-2k}(u) \right\rangle = g(n-2k), \qquad (2.84)$$

Substituindo-se as Eqs.(2.83) e (2.84) na Eq.(2.82), tem-se

$$\begin{aligned} \left\langle f(x), \phi_{j+1,n}(x) \right\rangle = & \sum_{k=-\infty}^{\infty} h(n-2k) \left\langle f(x), \phi_{j,k}(x) \right\rangle \\ & + \sum_{k=-\infty}^{\infty} g(n-2k) \left\langle f(x), \psi_{j,k}(x) \right\rangle \end{aligned} \qquad (2.85)$$

Essa equação mostra que $A^d_{2^{j+1}} f$ pode ser reconstruída inserindo-se um zero entre cada amostra de $A^d_{2^j} f$ e $D_{2^j} f$ e convoluindo-se os resultados com os filtros H e G,

respectivamente. Na Fig.2.4, pode-se ver o diagrama de blocos do algoritmo de reconstrução.

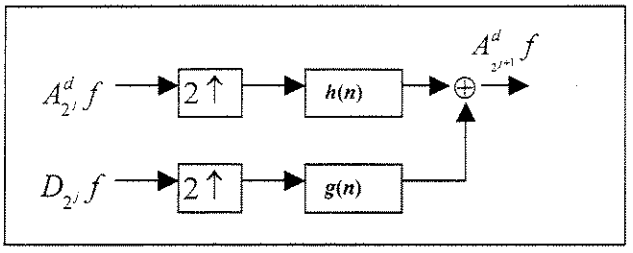


Fig. 2.4: Diagrama em blocos do esquema de reconstrução da representação *wavelet*

Observe que comparando-se os algoritmos de decomposição e síntese utilizam-se os filtros de resposta impulsiva $h(n)$ e $g(n)$, ao passo que na decomposição utilizam-se as versões invertidas $h(-n)$ e $g(-n)$ desses filtros.

As expressões (2.56), (2.64), (2.65) e (2.71) são as equações características de dois filtros conjugados em quadratura, com resposta de freqüência $G(\omega)$ e $H(\omega)$ [22].

V.6. *WAVELETS* DE DAUBECHIES, DE SUPORTE COMPACTO

Ingrid Daubechies, em 1987, propôs um método geral para construção de bases *wavelets* de suporte compacto baseado em uma série de restrições ao filtro $H(\omega)$. Essas restrições, expressas algebricamente, levam a $H(\omega)$ gerando uma análise de multiresolução. Todos os detalhes sobre esse método podem ser encontrados nas referências [4, 16].

Qualquer resposta impulsiva com duração finita de um filtro espelhado fornece uma função escala de suporte compacto $\phi(x)$ (Lema 2.3). É imediato da Eq.(2.58) que a *wavelet* $\psi(x)$ também apresenta um suporte compacto. Pode-se, entretanto, impor condições a esses filtros de forma a se garantir que as *wavelets* $\psi(x)$ sejam regulares.

V.6.1. ALGORITMO DE DAUBECHIES PARA A CONSTRUÇÃO DE FILTROS FIRS QUE GEREM *WAVELETS* REGULARES [16].

1. Escolha um número inteiro $N>0$
2. Escolha um polinômio ímpar $R(x)$, sujeito às restrições especificadas abaixo.
3. Faça

$$C(z) = \sum_{n=0}^{N-1} \binom{N-1+n}{n} \left(\frac{2 - z - z^{-1}}{4} \right)^n + \left(\frac{2 - z - z^{-1}}{4} \right)^N R\left(\frac{z + z^{-1}}{4} \right) \tag{2.86}$$

onde: $C(z)$ é um polinômio real simétrico em z e z^{-1}

O polinômio $R(x)$ deve ser escolhido de forma que $C(\omega)$ (que é real, dado a simetria de $C(z)$) seja não negativo para $\forall \omega$. $R(x) = 0$, por exemplo, satisfaz essa exigência.

4. Fatore $C(z)$ como

$$C(z) = D(z)D(z^{-1}), \tag{2.87}$$

onde: $D(z)$ é um polinômio em z.

Isso é possível de 2^K diferentes maneiras, onde K é o grau de $C(z)$. Por exemplo, pode-se tomar $D(z)$ como sendo o polinômio com raízes iguais às raízes de $C(z)$ que se encontram dentro do círculo de raio unitário.

5. Tome $H(z)$ como sendo:

$$H(z) = \left(\frac{1 + z^{-1}}{2} \right) D(z) \tag{2.88}$$

e $G(z)$ como na Eq.(2.71).

6. Construa $\phi(t)$ com um grau de precisão arbitrário utilizando a Eq.(2.52) de forma recursiva e $\psi(t)$ através da Eq.(2.72).

O parâmetro N do algoritmo de Daubechies controla a regularidade de $\phi(t)$ e $\psi(t)$. Utilizando-se o algoritmo dado e fazendo $R(z) = 0$ e $D(z)$ conforme sugerido no passo 4, pode-se obter uma família de *wavelets* ortogonais, regulares, de comprimento finito e com reconstrução exata, denominadas *wavelets* de Daubechies [16].

Quando $N = 1$ obtem-se, a *wavelet* de Haar [12]. Na Tabela 2.1 apresenta-se os coeficientes dos filtros passa-baixas das *wavelets* de Daubechies para $2 \leq N \leq 5$. Na Fig. 2.5 mostra-se os gráficos de ϕ e ψ para $N = 2, 3, 4$ [16]. Observe que as *wavelets* de Daubechies são assimétricas e que sua regularidade aumenta com N [4, 12, 16, 23].

Tabela 2.1: Coeficientes do filtro passa-baixas Daubechies para N =2,3,4.

	n	$h_N(n)$
$N=2$	0	0.482962913145.
	1	0.836516303738
	2	0.224143868042
	3	-0.129409522551
$N=3$	0	0.332670552950
	1	0.806891509311
	2	0.459877502118
	3	-0.135011020010
	4	-0.085441273882
	5	0.035226291882
$N=4$	0	0.230377813309
	1	0.714846570553
	2	0.630880767930
	3	-0.027983769417
	4	-0.187034811719
	5	0.030841381836
	6	0.032883011667
	7	-0.010597401785

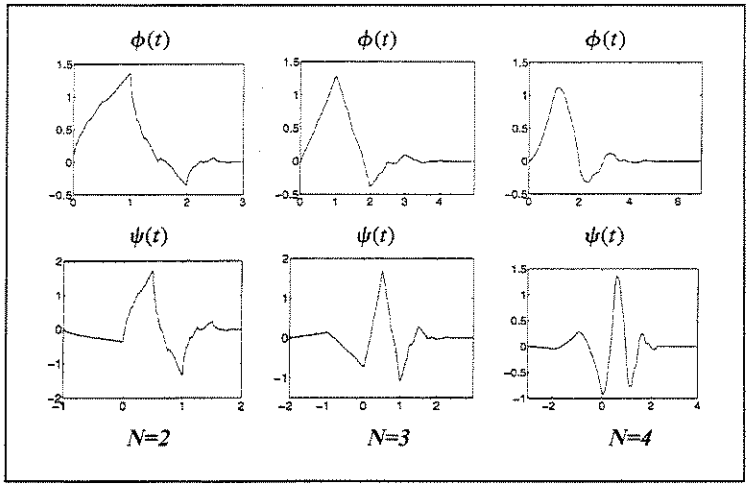


Fig. 2.5:Funções escalas ϕ e *wavelets* de Daubechies ψ para $N = 2,3,4$.

2.3. *WAVELETS* BIORTOGONAIS

I. INTRODUÇÃO

Projeções ortogonais permitem que um sinal seja decomposto em um conjunto de coeficientes independentes, onde cada elemento corresponde a um elemento dessa base ortogonal. Como já foi visto, esses coeficientes são calculados através da projeção da função em cada um dos elementos da base. Nesse tipo de representação não há redundância.

Em algumas aplicações, como, por exemplo, no processamento digital de imagens, é importante que a transformada utilizada apresente certas características como: simetria, regularidade e suporte compacto [24]. A regularidade das *wavelets* está relacionada com o tipo de função que a transformada pode representar. Quanto maior a regularidade, maior o número de sinais possíveis de serem representados com maior fidelidade. A simetria das *wavelets* é muito importante para o processamento de imagens, pois as imagens exigem a utilização de filtros de fase linear. Tais filtros permitem cascateamento de bancos de filtros sem a necessidade de compensação de fase. Além disso, os filtros de fase linear não propagam os erros de estágios individuais. O suporte compacto exige que os filtros apresentem resposta impulsiva de duração finita (FIR) e proporciona algoritmos eficientes. Os filtros utilizados no esquema de decomposição/reconstrução da transformada *wavelet* devem garantir a reconstrução perfeita do sinal, ou seja, nenhuma informação deve ser perdida.

A única *wavelet* ortogonal que obedece a todas essas exigências é a *wavelet* de Haar [4, 16]. Essa *wavelet*, entretanto não é contínua e, portanto, não pode ser utilizada em aplicações práticas. Para contornar esse problema, relaxa-se a condição de ortogonalidade, dando lugar à biortogonalidade. Existem *wavelets* biortogonais que atendem a todas as exigências citadas. Além disso, os filtros utilizados para sua implementação são bem mais flexíveis e fáceis de se projetar [16, 25, 26, 27].

II. *WAVELETS* BIORTOGONAIS. IMPLEMENTAÇÃO

Os filtros de reconstrução (h,g) e $(\tilde{h},\tilde{g})$ geram duas funções escala e *wavelet*, que no domínio do tempo satisfazem as seguintes relações,

$$\phi(t) = 2^{1/2}\sum_{n=-\infty}^{\infty} h(n)\phi(2t-n), \quad \tilde{\phi}(t) = 2^{1/2}\sum_{n=-\infty}^{\infty} \tilde{h}(n)\tilde{\phi}(2t-n), \tag{2.89}$$

$$\psi(t) = 2^{1/2}\sum_{n=-\infty}^{\infty} g(n)\phi(2t-n), \quad \tilde{\psi}(t) = 2^{1/2}\sum_{n=-\infty}^{\infty} \tilde{g}(n)\tilde{\phi}(2t-n). \tag{2.90}$$

Para uma reconstrução perfeita o filtro deve satisfazer as seguintes condições [11, 12, 16, 18, 27]

$$H^*(\omega)\tilde{H}(\omega) + H^*(\omega+\pi)\tilde{H}(\omega+\pi) = 2 \tag{2.91}$$

e

$$G(\omega) = e^{-j\omega}\tilde{H}^*(\omega+\pi), \quad \tilde{G}(\omega) = e^{-j\omega}H^*(\omega+\pi). \tag{2.92}$$

As funções *wavelets* devem ter média nula como já foi demonstrado, o que significa, $\Psi(0) = \tilde{\Psi}(0) = 0$. Isso é obtido a partir de , $G(0) = \tilde{G}(0) = 0$ e conseqüentemente, $H(\pi) = \tilde{H}(\pi) = 0$. Por outro lado, a condição de reconstrução perfeita expressa pela Eq.(2.91) implica, que $H^*(0)\tilde{H}(0) = 2$ e como uma conseqüência imediata, obtem-se $H(0) = \tilde{H}(0) = \sqrt{2}$ [12].

Supondo-se que h e $\tilde{h}$ são filtros de resposta impulsiva finita (FIR), demonstra-se em [12] que,

$$\Phi(\omega) = \prod_{p=1}^{\infty}\frac{H(2^{-p}\omega)}{2^{1/2}} \quad e \quad \tilde{\Phi}(\omega) = \prod_{p=1}^{\infty}\frac{\tilde{H}(2^{-p}\omega)}{2^{1/2}}, \tag{2.93}$$

são as transformadas de Fourier de funções de suporte compacto. Porém, essas funções podem exibir um comportamento não esperado e ter energia infinita. Condições adicionais devem ser impostas para garantir que Φ e $\tilde{\Phi}$ sejam as transformadas de Fourier de funções de energia finita. Os teoremas e demonstrações sobre as condições necessárias e suficientes para a construção de bases *wavelets* biortogonais podem ser encontrados em [12, 28, 29].

II.1. *WAVELETS* BIORTOGONAIS VS ORTOGONAIS

Biortogonalidade significa que para qualquer $(j,k,l,m) \in Z$,

$$\left\langle \psi_{j,k}(t), \widetilde{\psi}_{l,m}(t) \right\rangle = \int_{-\infty}^{\infty} \psi_{j,k}(t)\ \widetilde{\psi}_{l,m}(t)dt = \delta_{jl}\delta_{km}. \tag{2.94}$$

Segundo a teoria de *frames* apresentada na Seção 2.1, se uma *wavelet* ψ gera um *frame*, pode-se expressar qualquer função $f(x) \in L^2(R)$ através de qualquer uma das seguintes formas

$$f = \sum_{j,k} \left\langle f, \widetilde{\psi}_{j,k} \right\rangle \psi_{j,k} = \sum_{j,k} \left\langle f, \psi_{j,k} \right\rangle \widetilde{\psi}_{j,k}, \tag{2.95}$$

onde $\{\psi_{j,k}\}$ e $\{\widetilde{\psi}_{j,k}\}$ são *frames* duais ou bases *wavelets* duais, conforme definido na Eq.(2.28). Quando $\{\psi_{j,k}\}$ é exato, os conjuntos $\{\psi_{j,k}\}$ e $\{\widetilde{\psi}_{j,k}\}$ são biortogonais, ou seja cumprem com a condição expressa na Eq.(2.94).

Se os *frames* $\{\psi_{j,k}\}$ e $\{\widetilde{\psi}_{j,k}\}$ são estreitos e todos seus componentes têm norma unitária, as *wavelets* formam uma base ortogonal. Como conseqüência, $\widetilde{\psi}_{j,k} = \psi_{j,k}$ e a partir da Eq.(2.95) pode-se expressar

$$f = \sum_{j,k} \left\langle f, \psi_{j,k} \right\rangle \psi_{j,k}. \tag{2.96}$$

Pode-se então perceber que de fato as bases *wavelets* biortogonais são uma generalização das bases *wavelets* ortogonais.

II.1.1. PRINCIPAIS DIFERENÇAS ENTRE AS *WAVELETS* ORTOGONAIS E BIORTOGONAIS

- Os filtros ortogonais possuem necessariamente o mesmo comprimento, que deve ser par. Essa restrição não é imposta aos filtros biortogonais.
- A biortogonalidade permite que *wavelets* e funções escalas simétricas sejam construídas. Essa é uma das principais vantagens das *wavelets* biortogonais.
- A identidade de Parseval, não é válida para os sistemas biortogonais, ou seja, a norma da soma dos coeficientes não é idêntica à norma da função. Essa é uma das principais desvantagens das *wavelets* biortogonais.

2.4. MULTIRESOLUÇÃO BIORTOGONAL

Para a análise de multiresolução para *wavelets* biortogonais tem-se:

- Dois espaços de aproximação $\{V_{2^j}\}_{j\in Z}$ e $\{\widetilde{V}_{2^j}\}_{j\in Z}$;
- Um par de funções de escala duais ϕ e $\widetilde{\phi}$;
- Dois espaços de detalhamentos $\{W_{2^j}\}_{j\in Z}$ e $\{\widetilde{W}_{2^j}\}_{j\in Z}$;
- Um par de *wavelets* duais ψ e $\widetilde{\psi}$.

Como os subespaços de aproximação obedecem às propriedades apresentadas na Seção 2.2 cumprem, portanto à seguinte hierarquia,

$$\ldots \subset V_{2^{-1}} \subset V_{2^0} \subset V_{2^1} \subset V_{2^2} \subset V_{2^3} \subset \ldots, \tag{2.97}$$

$$\ldots \subset \widetilde{V}_{2^{-1}} \subset \widetilde{V}_{2^0} \subset \widetilde{V}_{2^1} \subset \widetilde{V}_{2^2} \subset \widetilde{V}_{2^3} \subset \ldots. \tag{2.98}$$

As bases $\{\phi_{j,k}\}_{j\in Z}$ e $\{\widetilde{\phi}_{j,k}\}_{j\in Z}$ geram os subespaços V_{2^j} e $\widetilde{V}_{2^j}$ respectivamente. $\phi_{j,k}$ e $\widetilde{\phi}_{j,k}$ são bases não-ortogonais construídas a partir de deslocamentos, dilatações e compressões das funções escalas ϕ e $\widetilde{\phi}$, respectivamente. Essas funções por sua vez, obedecem à seguinte relação

$$\langle \phi(t), \widetilde{\phi}(t-m) \rangle = \delta(m). \tag{2.99}$$

O subespaço W_{2^j} é o complemento não-ortogonal de V_{2^j} em $V_{2^{j+1}}$ e da mesma forma $\widetilde{W}_{2^j}$ é o complemento não-ortogonal de $\widetilde{V}_{2^j}$ em $\widetilde{V}_{2^{j+1}}$, ou seja,

$$V_{2^{j+1}} = V_{2^j} \oplus W_{2^j}, \qquad V_{2^j} \textit{ não é ortogonal à } W_{2^j}, \tag{2.100}$$

$$\widetilde{V}_{2^{j+1}} = \widetilde{V}_{2^j} \oplus \widetilde{W}_{2^j}, \qquad \widetilde{V}_{2^j} \textit{ não é ortogonal à } \widetilde{W}_{2^j}. \tag{2.101}$$

Os subespaços de aproximação W_{2^j} e $\widetilde{W}_{2^j}$ obedecem à seguinte hierarquia,

$$\ldots \subset W_{2^{-1}} \subset W_{2^0} \subset W_{2^1} \subset W_{2^2} \subset W_{2^3} \subset \ldots, \tag{2.102}$$

$$\ldots \subset \widetilde{W}_{2^{-1}} \subset \widetilde{W}_{2^0} \subset \widetilde{W}_{2^1} \subset \widetilde{W}_{2^2} \subset \widetilde{W}_{2^3} \subset \ldots. \tag{2.103}$$

Do mesmo modo, as bases $\{\psi_{j,k}\}_{j\in Z}$ e $\{\widetilde{\psi}_{j,k}\}_{j\in Z}$ geram os subespaços $\{W_{2^j}\}$ e $\{\widetilde{W}_{2^j}\}$ respectivamente. As bases $\psi_{j,k}$ e $\widetilde{\psi}_{j,k}$ também são não-ortogonais e a partir de deslocamentos, dilatações e compressões das funções *wavelets* ψ e $\widetilde{\psi}$, respectivamente. Essas funções por sua vez, obedecem à seguinte relação

$$\langle \psi(t), \widetilde{\psi}(t-m)\rangle = \delta(m).$$

Além dessas propriedades, a seguinte também é válida para a análise de multiresolução biortogonal [4]

$$V_{2^j} \perp \widetilde{W}_{2^j} \quad e \quad \widetilde{V}_{2^j} \perp W_{2^j},$$

que implica nas seguintes relações

$$\langle \phi(t), \widetilde{\psi}(t-m)\rangle = \langle \widetilde{\phi}(t), \psi(t-m)\rangle = \delta(m). \tag{2.104}$$

Sejam $\{\phi_{j,k}\}_{j\in Z}$ e $\{\widetilde{\phi}_{j,k}\}_{j\in Z}$ os dois *frames* duais geradores dos subespaços V_{2^j} e $\widetilde{V}_{2^j}$. De acordo com a teoria de *frames*, pode-se calcular a projeção de uma função $f(x) \in L(R^2)$ na base $\{\phi_{j,k}\}_{j\in Z}$. Então, a aproximação $f(x)$ no nível de resolução 2^j é dada pela seguinte expressão

$$A_{2^j} f(x) = \sum_{n=-\infty}^{\infty} \langle f(u), \widetilde{\phi}_{j,n}(u)\rangle \phi_{j,n}(x). \tag{2.105}$$

Da mesma forma, sejam $\{\psi_{j,k}\}$ e $\{\widetilde{\psi}_{j,k}\}$ os dois *frames* duais geradores dos subespaços $\{W_{2^j}\}$ e $\{\widetilde{W}_{2^j}\}$. Pode-se também calcular a projeção de uma função $f(x) \in L(R^2)$ na base $\{\psi_{j,k}\}_{j\in z}$ geradora de espaço W_{2^j}. Essa projeção, no nível de resolução 2^j, é dada pela seguinte expressão

$$P_{W_{2^j}} f(x) = \sum_{n=-\infty}^{\infty} \langle f(u), \widetilde{\psi}_{j,n}(u)\rangle \psi_{j,n}(x). \tag{2.106}$$

Como $\widetilde{\phi}_{j,k} \in \widetilde{V}_{2^j} \subset \widetilde{V}_{2^{j+1}}$, pode-se decompô-la em $\widetilde{V}_{2^{j+1}}$ e se chegar à seguinte expressão

$$\widetilde{\phi}_{j,n}(x) = \sum_{k=-\infty}^{\infty} \langle \widetilde{\phi}_{j,n}(u), \phi_{j+1,k}(u)\rangle \widetilde{\phi}_{j+1,k}(x) = \sum_{k=-\infty}^{\infty} \langle \widetilde{\phi}_{-1,0}(u), \phi_{0,k-2n}(u)\rangle \widetilde{\phi}_{j+1,k}(x). \tag{2.107}$$

Observe que a Eq.(2.107) pode ser interpretada como uma filtragem e assim, pode-se definir $\widetilde{h}(n)$ como sendo um filtro discreto com resposta impulsiva definida por

$$\tilde{h}(n) = \left\langle \tilde{\phi}_{-1,0}(u), \phi_{0,n}(u) \right\rangle . \tag{2.108}$$

Da mesma forma, como $\tilde{\psi}_{j,k} \in \tilde{W}_{2^j} \subset \tilde{V}_{2^{j+1}}$ pode-se decompô-la em $\tilde{V}_{2^{j+1}}$ e então obter a seguinte expressão

$$\tilde{\psi}_{j,n}(x) = \sum_{k=-\infty}^{\infty} \left\langle \tilde{\psi}_{j,n}(u), \phi_{j+1,k}(u) \right\rangle \tilde{\phi}_{j+1,k}(x) = \sum_{k=-\infty}^{\infty} \left\langle \tilde{\psi}_{-1,0}(u), \phi_{0,k-2n}(u) \right\rangle \tilde{\phi}_{j+1,k}(x) . \tag{2.109}$$

Observe que a Eq.(2.109) pode também ser interpretada como sendo uma filtragem e, assim, pode-se definir $\tilde{g}(n)$ como sendo um filtro discreto com resposta impulsiva definida por

$$\tilde{g}(n) = \left\langle \tilde{\psi}_{-1,0}(u), \phi_{0,n}(u) \right\rangle . \tag{2.110}$$

Substituindo-se a Eq.(2.108) na Eq.(2.110) e a Eq.(2.106) na Eq.(2.109), obtem-se

$$\tilde{\phi}_{j,n}(x) = \sum_{k=-\infty}^{\infty} \tilde{h}(k-2n) \tilde{\phi}_{j+1,k}(x) , \tag{2.111}$$

$$\tilde{\psi}_{j,n}(x) = \sum_{k=-\infty}^{\infty} \tilde{g}(k-2n) \tilde{\phi}_{j+1,k}(x) . \tag{2.112}$$

Convoluindo-se as Eqs.(2.111) e (2.112) com $f(x)$, obtem-se

$$\left\langle f(x), \tilde{\phi}_{j,n}(x) \right\rangle = \sum_{k=-\infty}^{\infty} \tilde{h}(k-2n) \left\langle f(x), \tilde{\phi}_{j+1,k}(x) \right\rangle \tag{2.113}$$

$$\left\langle f(x), \tilde{\psi}_{j,n}(x) \right\rangle = \sum_{k=-\infty}^{\infty} \tilde{g}(k-2n) \left\langle f(x), \tilde{\phi}_{j+1,k}(x) \right\rangle \tag{2.114}$$

Assim, substituindo-se a Eq.(2.113) na Eq.(2.105) e a Eq.(2.114) na Eq.(2.106), tem-se:

$$A_{2^j}^d f = \left(\left\langle f(x), \tilde{\phi}_{j,n}(x) \right\rangle \right)_{n \in z} \tag{2.115}$$

$$D_{2^j}^d f = \left(\left\langle f(x), \tilde{\psi}_{j,n}(x) \right\rangle \right)_{n \in z} \tag{2.116}$$

Observações:

- A aproximação discreta do sinal no nível de resolução 2^j é dada pela expressão (2.115).
- O sinal de detalhamento de $f(x)$ é dado pela expressão (2.116).

- A partir das expressões (2.115) e (2.116), pode-se calcular $A_{2^j}^d f$ e $D_{2^j}^d f$ através da convolução de $A_{2^{j+1}}^d f$ com os filtros $\tilde{h}$ e $\tilde{g}$, respectivamente, descartando-se uma amostra sim e outra não do resultado.
- As expressões (2.115) e (2.116) caracterizam o processo de decomposição da análise de multiresolução biortogonal.

De acordo com a teoria de *frames*, pode-se calcular a projeção da função $f(x)$ na base $\{\tilde{\phi}_{j+1,k}\}_{j\in Z}$,ou seja, de forma semelhante a que foi realizada na Eq.(2.105). A aproximação da função $f(x)$ no nível de resolução 2^{j+1} (um nível acima do que feito na Eq.(2.105)) é dada pela seguinte expressão

$$A_{2^{j+1}} f(x) = \sum_{n=-\infty}^{\infty} \left\langle f(u), \tilde{\phi}_{j+1,n}(u) \right\rangle \phi_{j+1,n}(x). \tag{2.116}$$

O subespaço W_{2^j} é o complemento não-ortogonal de V_{2^j} em $V_{2^{j+1}}$. Portanto, o espaço $V_{2^{j+1}}$ pode ser gerado pela base $\{\phi_{j,k}, \psi_{j,k}\}_{j\in Z}$ e $\phi_{j+1,k}$ ser, interpretada como a soma de duas filtragens. Define-se então $\tilde{h}(n)$ e $\tilde{g}(n)$ como sendo dois filtros digitais de resposta impulsiva,

$$h(n) = \left\langle \tilde{\phi}_{-1,0}(u), \phi(u-n) \right\rangle, \tag{2.117}$$

$$g(n) = \left\langle \tilde{\psi}_{-1,0}(u), \phi(u-n) \right\rangle. \tag{2.118}$$

Substituindo-se as Eqs.(2.117) e (2.118) na expressão de $\phi_{j+1,k}$, obtem-se

$$\phi_{j+1,n}(x) = \sum_{k=-\infty}^{\infty} h(n-2k)\phi_{j,k}(x) + \sum_{k=-\infty}^{\infty} g(n-2k)\psi_{j,k}(x), \tag{2.119}$$

Convoluindo-se a Eq.(2.119) com $f(x)$, obtem-se

$$\left\langle f(x), \phi_{j+1,n}(x) \right\rangle = \sum_{k=-\infty}^{\infty} h(n-2k)\left\langle f(x), \phi_{j,k}(x) \right\rangle + \sum_{k=-\infty}^{\infty} g(n-2k)\left\langle f(x), \psi_{j,k}(x) \right\rangle. \tag{2.120}$$

Assim, o processo de reconstrução da análise de multiresolução é caracterizado pela seguinte relação

$$A_{2^{j+1}}^d f(x) = A_{2^j}^d f(x) + D_{2^j}^d f(x). \tag{2.121}$$

Observações:

- A expressão (2.121) permite calcular a aproximação $A^d_{2^{j+1}} f$ através da convolução de $A^d_{2^j} f$ e $D^d_{2^j} f$ com os filtros H e G, respectivamente, interpolando-se o resultado por um fator de 2.
- A transformada *wavelet* ortogonal utiliza os mesmos filtros h e g, tanto para a reconstrução quanto para a decomposição. Conforme estudado nesta seção, no caso biortogonal, os filtros de reconstrução, h e g são diferentes dos filtros de decomposição, $\tilde{h}$ e $\tilde{g}$ [24].
- Os filtros correspondentes à *wavelet* biortogonal são mais flexíveis, fáceis de se projetar e podem ser simétricos [24].

Na Fig.2.7 mostra-se o diagrama em blocos do algoritmo de decomposição e síntese da transformada *wavelet* biortogonal.

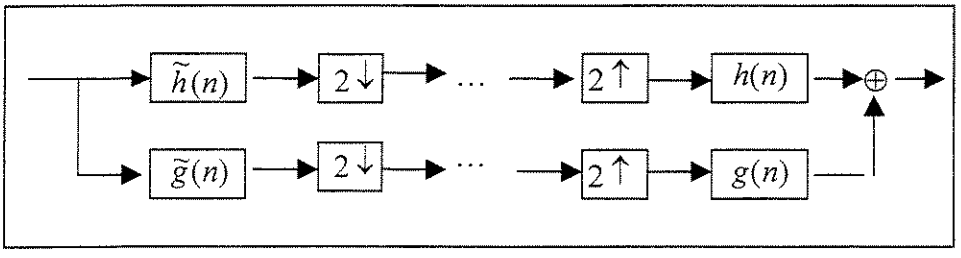


Fig. 2.7: Diagrama em blocos do algoritmo de decomposição e síntese da Transformada *wavelet* biortogonal.

I. Implementação das *Wavelets* Biortogonais

Nas referências [24, 30] realiza-se estudos matemáticos detalhados e são propostas algumas técnicas para a construção das bases *wavelets* biortogonais.

Em [24] provou-se que uma alta regularidade pode ser obtida, tanto para ψ quanto para $\tilde{\psi}$, desde que se selecione os filtros suficientemente longos. Em particular, se as funções ψ e $\tilde{\psi}$ forem, respectivamente, diferenciáveis $(k-1)$ e $(\tilde{k}-1)$, então as suas respostas em freqüências ($H(\omega)$ e $\tilde{H}(\omega)$) são divisíveis por $(1+e^{-j\omega})^k$ e $(1+e^{-j\omega})^{\tilde{k}}$, respectivamente, de tal forma que os filtros correspondentes ($h(n)$ e $\tilde{h}(n)$) tenham um comprimento maior que k e $\tilde{k}$, respectivamente.

A divisibilidade de $\widetilde{H}(\omega)$ por $(1+e^{-j\omega})^{\widetilde{k}}$ significa que ψ pode ter $\widetilde{k}$ momentos nulos consecutivos [4], ou seja,

$$\int x^l \psi(x)dx = 0, \qquad para\ l = 0,1,\ldots,\widetilde{k}-1. \tag{2.122}$$

É conhecido e pode ser provado utilizando-se as séries de Taylor [4], que se ψ tem $\widetilde{k}$ momentos nulos, então, os coeficientes $\langle f, \psi_{m,n} \rangle$ irão representar funções, $\widetilde{k}$ vezes diferenciáveis.

Muitos exemplos de *wavelets* biortogonais razoavelmente regulares podem ser construídos. Para as aplicações em vista neste trabalho, a regularidade de $\widetilde{\psi}_{m,n}$ que está diretamente ligada ao número de momentos de ψ, é mais importante que a regularidade de $\psi_{m,n}$ ou ao número de momentos nulos de $\widetilde{\psi}$. Dentro dos limites impostos pelo suporte compacto, procura-se escolher $\widetilde{k}$ o maior possível [24].

Em termos de $H(\omega)$ e $\widetilde{H}(\omega)$, a expressão de reconstrução perfeita é dada por [24]

$$H(\omega)\widetilde{H}(\omega) + H(\omega+\pi)\widetilde{H}(\omega+\pi) = 2. \tag{2.223}$$

Combinando-se as expressões (2.222) e (2.223) e levando-se em conta a imposição de divisibilidade de $H(\omega)$ e $\widetilde{H}(\omega)$ por $(1+e^{-j\omega})^{k}$ e $(1+e^{-j\omega})^{\widetilde{k}}$, chega-se, após algumas manipulações, à seguinte expressão [24]

$$H(\omega)\widetilde{H}(\omega) = \cos(\omega/2)^{2l}\left[\sum_{p=0}^{l-1}\binom{l-1+p}{p}\operatorname{sen}(\omega/2)^{2p} + \operatorname{sen}(\omega/2)^{2l}R(\omega)\right], \tag{2.224}$$

onde $R(\omega)$ é um polinômio ímpar em $\cos(\omega)$, e $2l = k + \widetilde{K}$ (o fato de os filtros h e $\widetilde{h}$ serem simétricos garante que $k+\widetilde{k}$ seja par).

A partir da Eq.(2.224) é possível se construir muitos exemplos de filtros. Na próxima seção, apresenta-se, em particular, três exemplos pertencentes a três diferentes famílias.

I.1. *WAVELETS SPLINE* BIORTOGONAIS COHEN-DAUBECHIES-FEAUVEAU

Se na Eq.(2.224) se faz $R(\omega) = 0$ e a resposta do filtro passa-baixas dual for definida pela seguinte expressão

$$\widetilde{H}(\omega) = \cos(\omega/2)^{\widetilde{k}}e^{-jk\omega}, \tag{2.225}$$

$$\text{onde} \begin{cases} k = 0, & se\ \tilde{k}\ é\ par \\ k = 1 & se\ \tilde{k}\ é\ impar \end{cases},$$

obtem-se a família de filtros conhecida como filtros *spline* conforme apresentado em [24], devido a que a função $\tilde{\phi}$ correspondente é *B-spline* ou filtro binomial como é apresentado em [30] onde os $\tilde{h}$ são simples coeficientes binomiais. Os filtros *spline* são simétricos, suaves e tem coeficientes diádicos. A resposta em freqüência do filtro passa-baixas é dada pela seguinte expressão

$$H(\omega) = \cos(\omega/2)^{2l-\tilde{k}}\, e^{jk\omega/2}\left[\sum_{p=0}^{l-1}\binom{l-1+p}{p}\operatorname{sen}(\omega/2)^{2p}\right]. \qquad (2.226)$$

Na Tabela 2.2 são apresentados os coeficientes h_n e $\tilde{h}_n$ dos filtros passa-baixas para $l = 3$ *e* $\tilde{k} = 2$. Observe que os filtros desse exemplo são simétricos com relação a 0, e possuem um comprimento ímpar. Essa é uma característica dos filtros *spline*.

Tabela 2.2: Coeficientes dos filtros *spline* para $l = 3$ e $\tilde{k} = 2$.

n	0	±1	±2	±3	±4
$2^{-1/2} h_n$	$45/64$	$19/64$	$-1/8$	$3/64$	$3/128$
$2^{-1/2} \tilde{h}_n$	$1/2$	$1/4$	0	0	0

I.2. VARIANTE DE *WAVELETS SPLINE* BIORTOGONAIS

Para se construir essa família de filtros faz-se $R(\omega) = 0$, na Eq.(2.224) e fatora-se o lado direito da expressão quebrando o polinômio de grau $l-1$ em $\operatorname{sen}(\omega/2)$,ou seja, em um produto de dois polinômios em $\operatorname{sen}(\omega/2)$. Na tentativa de tornar o comprimento do filtro h mais próximo do comprimento do filtro $\tilde{h}$, um dos dois polinômios resultantes, aquele que possui coeficientes reais é definido como H e outro como $\tilde{H}$.

O exemplo que é mostrado na Tabela 2.3 é o filtro mais curto dessa família. Corresponde a $l = k = 4$.

Tabela 2.3: Coeficientes do Filtro Variante de um *Spline* com comprimentos semelhantes para *l=k=4*.

n	0	± 1	± 2	± 3	± 4
$2^{-1/2} h_n$	0.602949	0.266864	-0.078223	-0.016864	0.026749
$2^{-1/2} \tilde{h}_n$	0.557543	0.295636	-0.028772	-0.045636	0

I.3. *WAVELETS* PRÓXIMAS ÀS ORTONORMAIS

Existem muitos exemplos de *wavelets* biortogonais para as quais $R(\omega) \neq 0$. Em particular existe uma escolha de $R(\omega)$ para a qual os filtros na Eq.(2.224) estão muito próximos e conseqüentemente, muito próximos a filtros *wavelets* ortonormais. Um desses casos é o filtro piramidal Laplaciano proposto em [31], para o qual $l = k = 2$ e

$$R(\omega) = 48\cos\left(\omega/2\right)175$$

Os coeficientes desse filtro podem ser consultados na Tabela 2.4.

Tabela 2.4: Coeficientes do Filtro Variante de um Filtro *Spline* com Comprimentos muito Semelhantes para *l=k=2*.

n	0	± 1	± 2	± 3	± 4
$2^{-1/2} h_n$	0.6	0.25	− 0.05	0	0
$2^{-1/2} \tilde{h}_n$	$17/28$	$73/280$	$-3/56$	$-3/280$	0

Os dois filtros biortogonais desse exemplo são ambos muito próximos de um filtro *wavelet* ortonormal de comprimento 6 apresentado em [32], onde eles foram denominados de *coiflets*. Sendo um filtro ortogonal, o *coiflet* não é simétrico. Os próximos (*l=k=4*) filtros de h e $\tilde{h}$ dessa família possuem comprimentos 9 e 15 e ambos são próximos ao *coiflet* de comprimento 12.

2.5. TRANSFORMADA *WAVELET* BIDIMENSIONAL

I. INTRODUÇÃO

A teoria *wavelet* pode ser estendida para qualquer dimensão $n > 1$ [33]. Para se definir a transformada *wavelet* bidimensional, considera-se que a *wavelet* bidimensional $\psi(x,y)$:

- É uma função oscilatória, de curta duração e de energia finita, ou seja,

$$\int_{-\infty-\infty}^{\infty}\int^{\infty}\psi(x,y)dxdy < \infty \tag{2.226}$$

- Satisfaz a propriedade de admissibilidade para o caso bidimensional, definida como,

$$C_\psi = \int_{-\infty}^{\infty}\frac{\left|\Psi(a^{-1}\omega_x, a^{-1}\omega_y)\right|^2}{a^{-1}}da < \infty, \qquad \forall(\omega_x,\omega_y)\in R^2, \tag{2.227}$$

onde $\Psi(\omega_x,\omega_y)$ é a transformada de Fourier de $\psi(x,y)$.

Define-se então a transformada *wavelet* contínua bidimensional como sendo,

$$TWC(a;b_x,b_y) = \int_{-\infty-\infty}^{\infty}\int^{\infty} f(x,y)\psi_{a;b_x,b_y}(x,y)dxdy, \tag{2.228}$$

onde $\Psi_{a;b_x,b_y}(x,y)$ são as *wavelets*-filhotes geradas a partir da *wavelet*-mãe $\psi(x,y)$ a partir da seguinte relação

$$\psi_{a;b_x,b_y}(x,y) = a^{-1}\psi\left(\frac{x-b_x}{a},\frac{y-b_y}{a}\right). \tag{2.229}$$

A TWC bidimensional apresenta as mesmas propriedades da TWC unidimensional, como por exemplo, a conservação da energia dada na Eq.(2.230). Assim como no caso unidimensional a relação é obtida a partir do teorema de Parseval

$$\int_{-\infty-\infty-\infty}^{\infty}\int\int\left|TWC(a;b_x,b_y)\right|^2 a^{-1}dadb_xdb_y = C_\psi\int_{-\infty-\infty}^{\infty}\int\left|f(x,y)\right|^2 dxdy. \tag{2.230}$$

Se $\psi(x,y)$ satisfaz a condição de admissibilidade dada pela Eq.(2.227), então existe uma TWC inversa definida como

$$f(x,y) = \frac{1}{C_\psi}\int_{-\infty-\infty}^{\infty}\int^{\infty} CWT(a;b_x,b_y)\,a^{-1}\psi\left(\frac{x-b_x}{a},\frac{y-b_y}{a}\right)dadb_xdb_y. \tag{2.231}$$

A transformada *wavelet* discreta bidimensional é semelhante à transformada unidimensional. Fazendo-se,

$$a = a_0^j, \quad b_x = nb_0a_0^j = k_xa_0^j, \quad e \quad b_y = mb_0a_0^j = k_ya_0^j,$$

então a TWD é ***definida*** por

$$TWD(j;k_x,k_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)a^{-1}\psi^*(a_0^{-j}x - k_x, a_0^{-j}y - k_y)dxdy\,. \tag{2.232}$$

As *wavelets* diádicas ($a_0 = 2$) bidimensionais são definidas pela seguinte expressão

$$\psi_{j;k_x,k_y}(x,y) = 2^{-j}\psi(2^{-j}x - k_x, 2^{-j}y - k_y)\,. \tag{2.233}$$

II. MULTIRESOLUÇÃO BIDIMENSIONAL

Uma análise de multiresolução em $L^2(R^2)$ é formada por uma seqüência de subespaços em $L^2(R^2)$ que satisfaz às propriedades que são mostradas na Seção 2.2, para o caso bidimensional.

Os resultados que são apresentados nesta seção são obtidos com a utilização de *wavelets* ortogonais, mas podem ser facilmente estendidos para casos não–ortogonais, como por exemplo, para *wavelets* biortogonais conforme é mostrado em [12, 18].

II.1. APROXIMAÇÃO POR MULTIRESOLUÇÃO EM $L^2(R^2)$

Seja $\left(V_{2^j}^2\right)_{j\in Z}$ a seqüência de subespaços em $L^2(R^2)$. A aproximação do sinal $f(x,y)$ no nível de resolução 2^j é igual a sua projeção no espaço vetorial $V_{2^j}^2$. O Teorema 2.2 é válido para o caso bidimensional [13] e pode-se demonstrar que existe uma única função de escala $\phi(x,y)$ bidimensional cujos deslocamentos, dilatações e compressões geram uma base ortonormal para cada espaço $V_{2^j}^2$. Seja,

$$\phi_{j;k_x,k_y}(x,y) = 2^j\phi(2^{-j}x - k_x, 2^{-j}y - k_y),$$

A família de funções,

$$\left(2^j\phi\left(2^{-j}x - k_x, 2^{-j}y - k_y\right)\right)_{(k_x,k_y)\in Z},$$

é uma base ortonormal em $V_{2^j}^2$. Para uma representação por multiresolução em $L^2(R^2)$, existe apenas uma função $\phi(x,y)$.

Nesta seção estuda-se o caso particular da aproximação por multiresolução separável proposto por [13, 34]. Nesse método, cada espaço vetorial $V_{2^j}^2$ pode ser decomposto como um produto tensorial de dois subespaços idênticos em $L^2(R)$,

$$V_{2^j}^2 = V_{2^j}^1 \otimes V_{2^j}^1 . \tag{2.234}$$

As seqüências de espaços vetoriais $\left(V_{2^j}^2\right)_{j\in Z}$ formam uma aproximação por multiresolução em $L^2(R^2)$ se e somente se, $\left(V_{2^j}^1\right)_{j\in Z}$ é uma aproximação por multiresolução em $L^2(R)$. Prova-se em [35] que a função escala $\phi(x,y)$ é uma função separável e portanto pode ser expressa da seguinte forma

$$\phi(x,y) = \phi(x)\phi(y), \tag{2.235}$$

onde $\phi(x)$ é a função escala unidimensional relativa a $\left(V_{2^j}^1\right)_{j\in Z}$. Para a base ortogonal em $V_{2^j}^2$, pode-se expressar

$$\left(\phi_{j;k_x,k_y}(x,y)\right)_{(k_x,k_y)\in Z^2} = \left(\phi_{j,k_x}(x)\phi_{j,k_y}(y)\right)_{(k_x,k_y)\in Z^2} . \tag{2.236}$$

Para o processamento de imagens, a utilização de uma aproximação por multiresolução separável dá ênfase às direções verticais e horizontais da imagem, representada pelas variáveis x e y.

A projeção de uma função $f(x,y) \in L^2(R^2)$ no espaço $V_{2^j}^2$ é dada por

$$A_{2^j} f(x,y) = \sum_{k_x=-\infty}^{\infty} \sum_{k_y=-\infty}^{\infty} \left\langle f(u,v), \phi_{j;k_x,k_y}(u,v) \right\rangle \phi_{j;k_x,k_y}(x,y) . \tag{2.237}$$

Logo, a aproximação de um sinal $f(x,y)$ no nível de resolução 2^j é caracterizada pelo seguinte conjunto de produtos internos

$$A_{2^j}^d f = \left(\left\langle f(x,y), \phi_{j;k_x}(x) \quad \phi_{j,k_y}(y) \right\rangle \right)_{(k_x,k_y)\in Z^2} , \tag{2.238}$$

onde $A_{2^j}^d f$ é a aproximação discreta do sinal $f(x,y)$ no nível de resolução 2^j.

II.2. Implementação de uma Transformada de Multiresolução Bidimensional

Na prática, os sinais bidimensionais $f(x,y)$ com as quais se trabalha são imagens discretas com resolução máxima finita. Assim como no caso unidimensional, supõe-se [13],

- Resolução máxima igual 1.
- $A_{2^j}^d f(x,y)$ é a aproximação discreta nessa resolução máxima.
- Se $A_{2^j}^d f(x,y)$ tem N *pixels* pode-se mostrar que para $j<0$, a sua aproximação discreta com resolução $2^j\left(A_{2^j}^d f(x,y)\right)$ tem $2^j N$ *pixels*.
- Os possíveis problemas de bordas são amenizados supondo-se que a imagem original é simétrica em relação as bordas horizontais e verticais.

Seja $\left(V_{2^j}^2\right)_{j\in Z}$ uma aproximação por multiresolução bidimensional, e $\phi(x,y)$ a função de escala correspondente. Sabe-se que $\phi_{j;k_x,k_y}(x,y)$ pertence a $V_{2^j}^2 \subset V_{2^{j+1}}^2$. Logo, pode-se expandir a função $\phi_{j;k_x,k_y}(x,y)$ na base ortonormal $\phi_{j+1;k_x,k_y}(x,y)$ do espaço $V_{2^{j+1}}^2$,ou seja

$$\phi_{j;k_x,k_y}(x,y)=\sum_{k_x=-\infty}^{\infty}\sum_{k_y=-\infty}^{\infty}\left\langle\phi_{j;n_x,n_y}(u,v),\phi_{j+1;k_x,k_y}(u,v)\right\rangle\phi_{j+1;k_x,k_y}(u,v). \qquad (2.239)$$

Fazendo-se uma substituição de variáveis, obtem-se

$$\phi_{j;k_x,k_y}(x,y)=\sum_{k_x=-\infty}^{\infty}\sum_{k_y=-\infty}^{\infty}\left\langle\phi_{-1;0,0}(u,v),\phi_{0;(k_x-2n_x),(k_y-2n_y)}(u,v)\right\rangle\phi_{j+1;k_x,k_y}(u,v). \qquad (2.240)$$

A expressão anterior pode ser interpretada como sendo uma filtragem. Pode-se então definir H_2 como sendo um filtro bidimensional discreto de resposta impulsiva dada por

$$h_2(n,m)=\phi_{-1;0,0}(x,y),\phi_{0;n,m}(x,y)=\left\langle\phi_{-1;0}(x),\phi_{0;n}(x)\right\rangle\left\langle\phi_{-1;0}(y),\phi_{0;m}(y)\right\rangle=h(n)h(m) \quad (2.241)$$

Conforme foi definido na Eq.(2.45), a resposta impulsiva h_2 pode ser decomposta no produto de dois filtros digitais unidimensionais de resposta impulsiva $h(.)$. Portanto, pode-se deduzir que o filtro H_2 é separável nas direções x e y.

Substituindo-se a Eq.(2.239) na Eq.(2.240) e convoluindo-se o resultado com $f(x,y)$ obtem-se

$$\left\langle f(x,y),\phi_{j;k_x,k_y}(x,y)\right\rangle=\sum_{k_x=-\infty}^{\infty}h(k_y-2n_y)\sum_{k_y=-\infty}^{\infty}h(k_y-2n_y)\left\langle f(x,y),\phi_{j+1;k_x,k_y}(x,y)\right\rangle. \quad (2.242)$$

Comparando-se as Eqs.(2.242) e (2.241) pode-se concluir que [13]:

- $A_{2^j}^d f$ pode ser calculada através de filtragens sucessivas de $A_{2^{j+1}}^d f$ em ambas as direções.

- Inicialmente filtra-se $A^d_{2^{j+1}} f$ na direção x (linhas) com um filtro unidimensional $h(.)$, descartando-se uma amostra sim e outra não nessa direção, ou seja, uma linha sim e outra não.
- O resultado é filtrado na direção y (colunas) com o mesmo filtro unidimensional, descartando-se uma amostra sim e outra não, nessa direção, ou seja, uma coluna sim e outra não.

II.3. O SINAL DE DETALHAMENTO EM $L^2(R^2)$

No caso bidimensional o sinal de detalhamento no nível de resolução 2^j, é igual à projeção ortogonal do sinal no subespaço $W^2_{2^j}$, que é o complemento ortogonal de $V^2_{2^j}$ em $V^2_{2^{j+1}}$. O Teorema 2.5 que apresenta a continuação é uma extensão simples do Teorema 2.2. De acordo com esse último, pode-se construir uma base ortonormal $W^2_{2^j}$ através do escalonamento e deslocamento das três funções *wavelets*, $\psi^1(x,y), \psi^2(x,y)$ *e* $\psi^3(x,y)$ [13].

Teorema 2.5: Seja $\left(V^2_{2^j}\right)_{j\in Z}$ uma aproximação por multiresolução separável em $L^2(R^2)$. Seja $\phi(x,y) = \phi(x)\phi(y)$ a função escala bidimensional separável e $\psi(x)$ a *wavelet* unidimensional associada a função escala $\phi(x)$. Então, as três;

$$\psi^1(x,y) = \phi(x)\psi(y), \quad \psi^2(x,y) = \psi(x)\phi(y), \quad \psi^3(x,y) = \psi(x)\psi(y) \tag{2.243}$$

são tais que,

$$\left(2^j\psi^1(2^j x - k_x, 2^j y - k_y),\ 2^j\psi^2(2^j x - k_x, 2^j y - k_y) \text{ e } 2^j\psi^3(2^j x - k_x, 2^j y - k_y)\right)_{(k_x,k_y)\in Z^2} \tag{2.244}$$

é uma base ortonormal em $W^2_{2^j}$.

A diferença de informação entre $A^2_{2^j+1} f$ *e* $A^2_{2^j} f$ é igual à projeção ortogonal de $f(x,y)$ em $W^2_{2^j}$. A partir do Teorema 2.5 pode-se afirmar que a diferença de informação é dada pelas três imagens de detalhes,

$$D^1_{2^j} f = \left\langle\left\langle f(x,y), 2^j \psi^1(2^j x - k_x, 2^j y - k_y)\right\rangle\right\rangle_{(k_x,k_y)\in Z^2}$$
$$D^2_{2^j} f = \left\langle\left\langle f(x,y), 2^j \psi^2(2^j x - k_x, 2^j y - k_y)\right\rangle\right\rangle_{(k_x,k_y)\in Z^2} \quad (2.245), (2.246), (2.247).$$
$$D^3_{2^j} f = \left\langle\left\langle f(x,y), 2^j \psi^3(2^j x - k_x, 2^j y - k_y)\right\rangle\right\rangle_{(k_x,k_y)\in Z^2}$$

Devido ao fato de que as três *wavelets* são separáveis, as Eqs.(2.245), (2.246) e (2.247) podem ser expressas por

$$D^1_{2^j} f = \left(f(x,y) * 2^{j/2}\phi(2^j x - k_x)\ 2^{j/2}\psi(2^j y - k_y)\right)_{(k_x,k_y)\in Z^2}$$
$$D^2_{2^j} f = \left(f(x,y) * 2^{j/2}\psi(2^j x - k_x)\ 2^{j/2}\phi(2^j y - k_y)\right)_{(k_x,k_y)\in Z^2} \quad (2.248), (2.249), (2.250)$$
$$D^3_{2^j} f = \left(f(x,y) * 2^{j/2}\psi(2^j x - k_x)\ 2^{j/2}\psi(2^j y - k_y)\right)_{(k_x,k_y)\in Z^2}$$

III. Implementação de uma Representação *Wavelet* Bidimensional Ortogonal

Nesta seção mostra-se que em duas dimensões, $A^d_{2^j} f$ *e* $D^i_{2^j} f$ podem ser calculados com filtragem separável do sinal nas direções da abscissa e da ordenada [13].

Seja $\left(\psi^i_{j;k_x,k_y}(x,y)\right)_{1\le i\le 3}$ um conjunto de *wavelets* separáveis, conforme foi definido no Teorema 2.5 .Cada *wavelet* $\psi^i_{j;k_x,k_y}(x,y)$ pertence ao espaço $W^2_{2^j} \subset V^2_{2^{j+1}}$. Logo, pode-se expandir $\psi^i_{j;k_x,k_y}(x,y)$ na base ortonormal $\phi_{j+1;k_x,k_y}(x,y)$ do espaço $V^2_{2^{j+1}}$, ou seja

$$\psi^i_{j;k_x,k_y}(x,y) = \sum_{k_x=-\infty}^{\infty}\sum_{k_y=-\infty}^{\infty} \left\langle \psi^i_{j;k_x,k_y}(u,v), \phi_{j+1;k_x,k_y}(u,v)\right\rangle \phi_{j+1;k_x,k_y}(x,y). \quad (2.251)$$

Fazendo-se uma substituição de variáveis, obtem-se

$$\psi^i_{j;k_x,k_y}(x,y) = \sum_{k_x=-\infty}^{\infty}\sum_{k_y=-\infty}^{\infty} \left\langle \psi^i_{-1;0,0}(u,v), \phi_{0;k_x-2n_x,k_y-2n_y}(u,v)\right\rangle \phi_{j+1;k_x,k_y}(x,y). \quad (2.252)$$

A Eq.(2.251) pode ser interpretada como uma filtragem, onde G^i_2 é um filtro discreto bidimensional de resposta impulsiva

$$g^i(n,m) = \left\langle \psi^i_{-1;0,0}(u,v), \phi_{0;n,m}(u,v)\right\rangle, \quad (2.253)$$

onde *i* especifica o índice da *wavelet* utilizada na Eq.(2.253), de acordo com o que foi definido na Eq.(2.243). Para cada *wavelet i* existe um filtro correspondente de resposta

impulsiva $g^i(n,m)$. Assim, para $1 \le i \le 3$, os seguintes filtros bidimensionais são definidos

$$g^1(n,m) = \left\langle \phi_{-1,0}(u)\psi_{-1,0}(v), \phi_{0,n}(u)\phi_{0,m}(v) \right\rangle = \left\langle \phi_{-1,0}(u), \phi_{0,n}(u) \right\rangle \left\langle \psi_{-1,0}(v), \phi_{0,m}(v) \right\rangle = h(n)g(m) \quad (2.254)$$

$$g^2(n,m) = \left\langle \psi_{-1,0}(u)\phi_{-1,0}(v), \phi_{0,n}(u)\phi_{0,m}(v) \right\rangle = \left\langle \psi_{-1,0}(u), \phi_{0,n}(u) \right\rangle \left\langle \phi_{-1,0}(v), \phi_{0,m}(v) \right\rangle = g(n)h(m) \quad (2.255)$$

$$g^3(n,m) = \left\langle \psi_{-1,0}(u)\psi_{-1,0}(v), \phi_{0,n}(u)\phi_{0,m}(v) \right\rangle = \left\langle \psi_{-1,0}(u), \phi_{0,n}(u) \right\rangle \left\langle \psi_{-1,0}(v), \phi_{0,m}(v) \right\rangle = g(n)g(m) \quad (2.256)$$

Observa-se que:

- As respostas impulsivas dos filtros bidimensionais dados nas Eqs.(2.254), (2.255) e (2.256) podem ser decompostas em produtos das respostas impulsivas dos filtros unidimensionais h (.) e g (.).
- Esses filtros bidimensionais são separáveis e resultam em produtos de filtros unidimensionais que operam em duas direções distintas (x e y).

Convoluindo-se a Eq.(2.252) com f(x,y), e substituindo-se as expressões (2.254), (2.255) e (2.256) no resultado, obtem-se:

$$\left\langle f(x,y), \psi^1_{j;k_x,k_y}(x,y) \right\rangle = \sum_{k_x=-\infty}^{\infty} g(k_y - 2n_y) \sum_{k_y}^{\infty} h(k_x - 2n_x) \left\langle f(x,y), \psi^1_{j+1;k_x,k_y}(x,y) \right\rangle \quad (2.257)$$

$$\left\langle f(x,y), \psi^2_{j;k_x,k_y}(x,y) \right\rangle = \sum_{k_x=-\infty}^{\infty} h(k_y - 2n_y) \sum_{k_y}^{\infty} g(k_x - 2n_x) \left\langle f(x,y), \psi^2_{j+1;k_x,k_y}(x,y) \right\rangle \quad (2.258)$$

$$\left\langle f(x,y), \psi^3_{j;k_x,k_y}(x,y) \right\rangle = \sum_{k_x=-\infty}^{\infty} g(k_y - 2n_y) \sum_{k_y}^{\infty} g(k_x - 2n_x) \left\langle f(x,y), \psi^3_{j+1;k_x,k_y}(x,y) \right\rangle \quad (2.259)$$

Comparando-se as Eqs.(2.248), (2.249) e (2.250) com as Eqs.(2.257), (2.258) e (2.259), conclue-se que:

- As diferentes imagens de detalhes, $D^i_{2^j} f$, podem ser calculadas através de filtragens sucessivas com os filtros unidimensionais em ambas as direções.
- Inicialmente, $A^d_{2^{j+1}} f$, é filtrado na direção x (linhas), com um filtro unidimensional, descartando-se uma amostra sim e outra não nessa direção. Em seguida, o resultado é filtrado na direção y (colunas), com o outro filtro unidimensional.

- O filtro utilizado em cada direção, $g(.)$ $h(.)$, vai depender do valor de i conforme definido pelas expressões (2.257), (2.258) e (2.259).

IV. Algoritmo de Decomposição de uma Imagem

A representação *wavelet* bidimensional pode ser calculada, através do cascateamento de dois algoritmos piramidais unidimensionais. O algoritmo unidimensional, inicialmente, é aplicado nas linhas da imagem e depois nas colunas. A cada passo, a transformada *wavelet* bidimensional descompõe $A^d_{2^{j+1}}f$ em $A^d_{2^j}f, D^1_{2^j}f, D^1_{2^j}f$. Esse algoritmo é ilustrado pelo diagrama de blocos apresentado na Fig. 2.8.

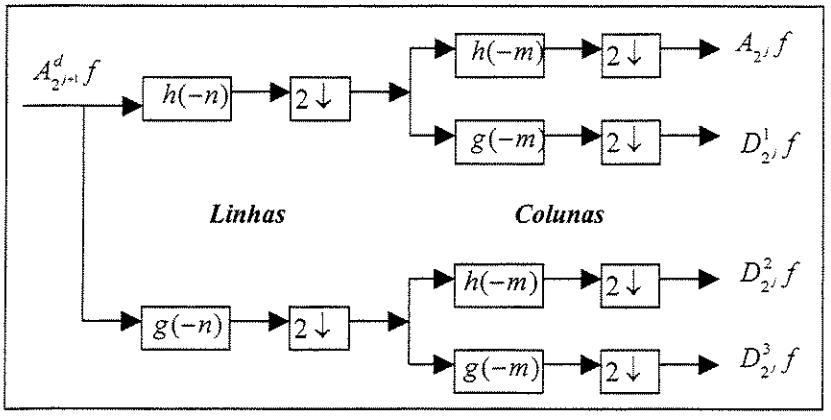


Fig. 2.8: Diagrama do Algoritmo de decomposição de uma imagem.

O algoritmo consiste em se realizar a filtragem das linhas de $A^d_{2^{j+1}}f$ com um filtro unidimensional, descartando-se uma linha sim e outra não. Em seguida, realiza-se uma filtragem das colunas do sinal resultante com outro filtro unidimensional, descartando-se uma coluna sim e outra não. Os filtros unidimensionais utilizados nessa decomposição são os filtros conjugados em quadratura $\widetilde{H}$ e $\widetilde{G}$ que possuem respostas impulsivas $h(-n)$ e $g(-n)$, respectivamente [13, 18].

Tabela 2.5: Relação entre as imagens decompostas e as freqüências de $f(x,y)$.

IMAGEM	CORRESPONDE ÀS FREQÜÊNCIAS.
$A^d_{2^J} f$	Mais baixas
$D^1_{2^J} f$	Verticais altas
$D^2_{2^J} f$	Horizontais altas
$D^3_{2^J} f$	Altas em ambas às direções.

Na Tabela 2.5 mostra-se a relação entre as imagens decompostas e suas correspondentes freqüências na imagem original. O arranjo das imagens resultantes para três estágios pode ser visto na Fig. 2.8.

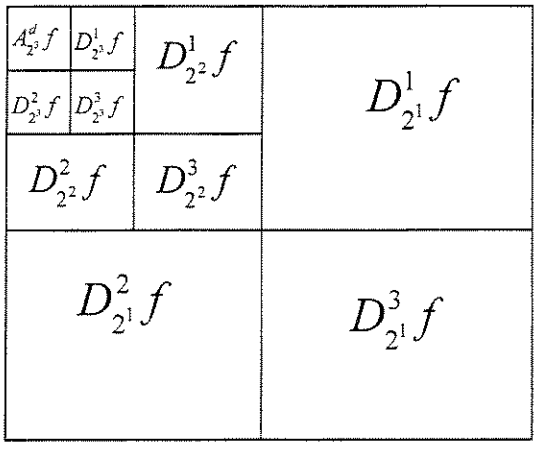


Fig. 1.9: Disposição das Imagens resultantes de uma decomposição através da Transformada *wavelet*.

Para qualquer $J>0$, uma imagem $A^d_1 f$ é completamente representada pelas $3J+1$ imagens discretas,

$$\left(A^d_{2^{-J}} f\left(D^1_{2^j} f\right)_{-J\le j\le -1}, \left(D^2_{2^j} f\right)_{-J\le j\le -1}, \left(D^3_{2^j} f\right)_{-J\le j\le -1}\right) \qquad (2.260)$$

Esse conjunto de imagens é denominado representação *wavelet* ortogonal em duas dimensões. A imagem $A^d_{2^{-J}} f$ é a aproximação de $A^d_1 f$ com nível de resolução 2^{-J} e $D^k_{2^j} f$ são as imagens de detalhamento para as diferentes orientações e nível de resoluções. Se a imagem original tem N *pixels*, então cada imagem $A^d_{2^J} f$, $D^1_{2^J} f$, $D^2_{2^J} f$ e $D^3_{2^J} f$ tem $2^J N$ *pixels* ($j<0$). O número total de *pixels* dessa nova representação é igual

ao número de *pixels* da imagem original. Por causa da ortogonalidade o volume de dados da representação não é aumentado [13].

V. RECONSTRUÇÃO DO SINAL

A imagem decomposta pode ser reconstruída utilizando-se a transformada *wavelet* 2-D. Como $W_{2^j}^2$ é o complemento ortogonal de $V_{2^j}^2$ em $V_{2^{j+1}}^2$, $\left(\phi_{j;k_x,k_y}(x,y), \left(\psi_{j;k_x,k_y}^i(x,y)\right)_{1\le i\le 3}\right)$ é a base ortonormal de $V_{2^{j+1}}^2$ e a função $\phi_{j;k_x,k_y}(x,y)$ pode ser decomposta da seguinte forma

$$\begin{aligned}\phi_{j+1;k_x,k_y}(x,y) = &\sum_{k_x=-\infty}^{\infty}\sum_{k_y=-\infty}^{\infty}\left\langle \phi_{j+1;n_x,n_y}(u,v), \phi_{j;k_x,k_y}(u,v)\right\rangle \phi_{j;k_x,k_y}(x,y) \\ &+\sum_{i=1}^{3}\sum_{k_x=-\infty}^{\infty}\sum_{k_y=-\infty}^{\infty}\left\langle \phi_{j+1;n_x,n_y}(u,v), \psi_{j;k_x,k_y}^i(u,v)\right\rangle \psi_{j;k_x,k_y}^i(u,v)\end{aligned} \quad . \quad (2.261)$$

Convoluindo-se a Eq.(2.261) com $f(x,y)$ e realizando-se algumas substituições, obtem-se

$$\begin{aligned}\left\langle f(x,y), \phi_{j+1;k_x,k_y}(x,y)\right\rangle = &\sum_{k_x=-\infty}^{\infty} h(2n_y-k_y)\sum_{k_x=-\infty}^{\infty} h(2n_x-k_x)\left\langle f(x,y), \phi_{j;k_x,k_y}(x,y)\right\rangle \\ &+\sum_{k_x=-\infty}^{\infty} g(2n_y-k_y)\sum_{k_x=-\infty}^{\infty} h(2n_x-k_x)\left\langle f(x,y), \psi_{j;k_x,k_y}^1(x,y)\right\rangle \\ &+\sum_{k_x=-\infty}^{\infty} h(2n_y-k_y)\sum_{k_x=-\infty}^{\infty} g(2n_x-k_x)\left\langle f(x,y), \psi_{j;k_x,k_y}^2(x,y)\right\rangle \\ &+\sum_{k_x=-\infty}^{\infty} g(2n_y-k_y)\sum_{k_x=-\infty}^{\infty} g(2n_x-k_x)\left\langle f(x,y), \psi_{j;k_x,k_y}^3(x,y)\right\rangle\end{aligned} \quad .(2.262)$$

O diagrama de blocos desse algoritmo é ilustrado na Fig. 2.10. Comparando-se a Eq.(2.262) com as Eqs.(2.238), (2.248), (2.249) e (2.250), conclue-se que:

- A cada passo a imagem $A_{2^{j+1}}^d f$ é reconstruída a partir de filtragens unidimensionais das imagens $A_{2^j}^d f, D_{2^j}^1 f, D_{2^j}^2 f$ *e* $D_{2^j}^3 f$.
- Primeiramente a coluna das imagens $A_{2^j}^d f, D_{2^j}^1 f, D_{2^j}^2 f$ *e* $D_{2^j}^3 f$ é interpolada com uma coluna de zeros e realiza-se a convolução das linhas resultantes com um filtro unidimensional.

➢ A seguir cada linha da imagem é interpolada com uma linha de zeros e realiza-se a convolução das colunas resultantes com um filtro unidimensional.

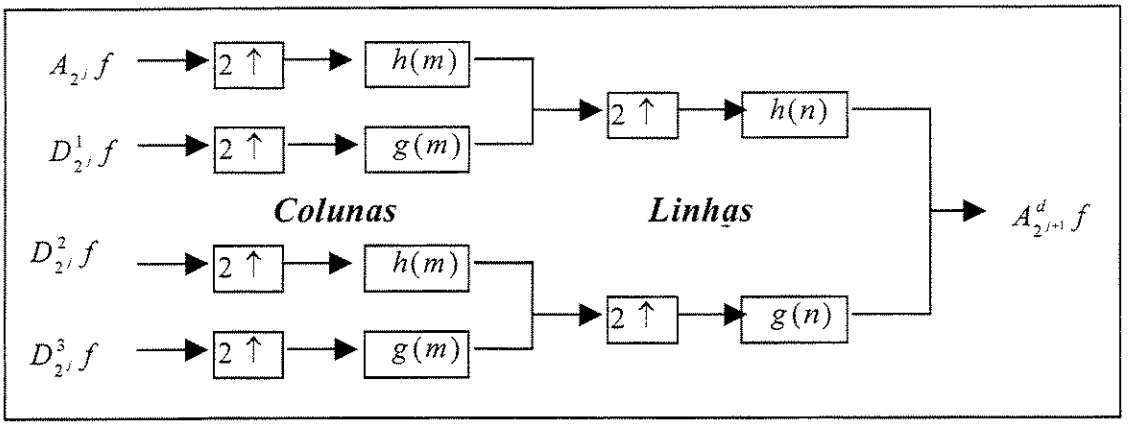


Fig. 2.10: Diagrama do algoritmo de reconstrução de uma imagem.

Os filtros unidimensionais utilizados nesse processo de reconstrução, são filtros conjugados em quadratura H e G com respostas impulsivas $h(n)$ $g(n)$. A imagem $A_1^d f$ é reconstruída a partir da sua transformada *wavelet*, repetindo-se o processo para $-J \leq j \leq -1$ [13].

2.6. SIMULAÇÕES E TESTES.

No Anexo A, são apresentadas partes das experiências práticas realizadas que permitem avaliar de forma comparativa os conceitos desenvolvidos neste capitulo. As simulações foram implementadas utilizando-se o programa "*Mathematica*" 4.0.

CAPÍTULO 3.

3.1. A TRANSFORMADA *WAVELET* NO PROCESSAMENTO DE IMAGENS

I. INTRODUÇÃO

As aplicações em processamento de imagens exigem que a transformada e, portanto os filtros responsáveis pela sua implementação, apresentem certas características, algumas das quais são,

1. ***Suporte compacto***: Se a função de escala e a *wavelet* têm suporte compacto, isto é, são funções não-nulas em um intervalo finito, as respostas impulsivas dos filtros correspondentes, têm comprimentos finitos (FIR). Assim, o esforço computacional da implementação é inferior a aquele exigido por funções que não apresentam essa característica.
2. ***Coeficientes racionais***: Quando se utilizam filtros, com coeficientes racionais ou melhor ainda, racionais diádicos, as operações de ponto flutuante são evitadas e um número menor de erros é introduzido.
3. ***Filtros de fase linear***: os filtros FIR utilizados devem ter fase linear (*wavelets* simétricas) evitando assim, problemas nas bordas das imagens e simplificando a implementação.
4. ***Suavidade ou Regularidade***: A compressão é obtida anulando-se coeficientes da transformada que apresentem amplitude inferior a um determinado valor. Numa compressão de imagens, se a *wavelet* não é suave, o erro devido a sua eliminação é mais facilmente detectado (visualmente). Além disso, um maior grau de suavidade corresponde a uma melhor localização em freqüência, dos filtros.
5. ***Comprimentos dos filtros***: Obviamente, sob o aspecto da implementação, é preferível que os filtros tenham comprimentos menores. Entretanto, há um compromisso entre o comprimento dos filtros, a suavidade e o número de momentos desvanecentes da *wavelet* e da função de escala. Quanto maior a suavidade e o número de momentos desvanecentes da *wavelet*, maior o comprimento dos filtros correspondentes.

Como as imagens são, na sua maioria suaves, parece apropriado utilizar *wavelets* de reconstrução exata ortogonais com boa regularidade. Infelizmente, nem todas essas condições anteriores podem ser satisfeitas simultaneamente, pois não existem filtros FIR de fase linear ortonormais de reconstrução exata. As *wavelets* que atendem à maioria dessas exigências simultaneamente são as *wavelets* biortogonais, as quais têm apresentado um desempenho interessante na compressão de imagens [24].

II. Compressão de Imagens

Compressão de imagem é o processo de redução da quantidade de bits necessária para representar uma imagem. A compressão em geral é possível devido ao fato de que o número de bits realmente necessários para se representar a informação contida na imagem original pode ser reduzido, resultado da redundância natural que a imagem apresenta.

A redundância pode ser definida matemática e quantitativamente da maneira definida a seguir. Suponha que n_1 e n_2 sejam os números de unidades de informação correspondentes a dois conjuntos de dados, que representam a mesma fonte de informação. Por exemplo, em processamento de imagens, suponha que n_1 representa o número de bits da imagem original e n_2 representa o número de bits da imagem comprimida. Desse modo, a redundância de dados relativa a n_1 é representada por $R_D = 1 - \frac{1}{C_R}$, onde $C_R = \frac{n_1}{n_2}$ e é denominada de taxa de compressão relativa.

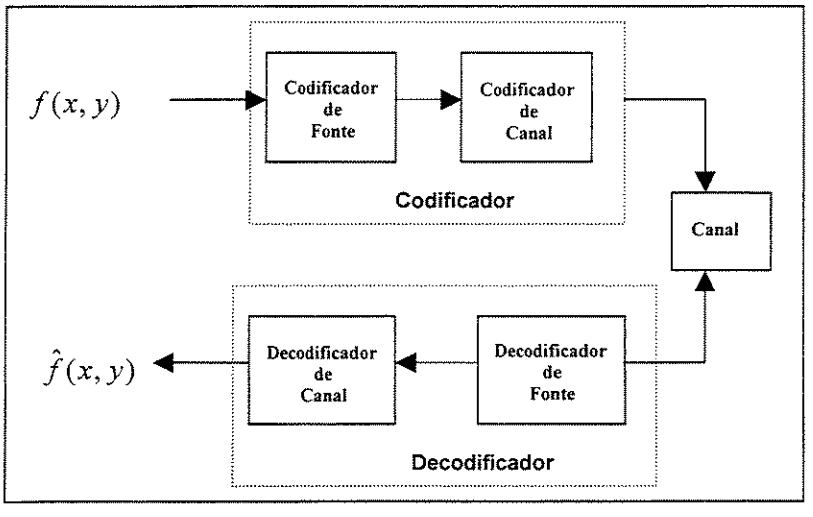


Fig. 3.1: Modelo de um sistema de compressão genérico.

Na Fig.3.1, mostra-se o diagrama em blocos de um sistema de compressão. O sistema pode ser dividido em dois blocos básicos, um codificador e um decodificador. O sinal de entrada $f(x,y)$, alimenta o codificador que cria um conjunto de símbolos. No caso de uma imagem, os símbolos representam os níveis de amplitude de cada *pixel* da imagem. Após a transmissão através do canal, a representação codificada é alimentada no decodificador, onde a imagem, reconstruída $\hat{f}(x,y)$ é gerada. Se $f(x,y)=\hat{f}(x,y)$, o sistema é livre de erros e a compressão é sem-perdas. Caso contrário a reconstrução é com-perdas e algum nível de distorção estará presente na imagem reconstruída.

A compressão sem-perdas apresenta a vantagem de não introduzir erros. Por outro lado, em geral, pouca compressão pode ser obtida dessa forma.

A imagem reconstruída na compressão com-perdas contém degradações em relação à imagem original. Um nível de compressão muito maior pode ser obtido, ao custo de uma maior degradação. É importante saber que essas degradações podem ou não ser visíveis, o que torna esse tipo de compressão muito atrativo.

III. Sistema Básico de Compressão Baseado na Transformada *Wavelet*.

O sistema pode ser dividido em três estágios: a transformada *wavelet*, a quantização e a codificação entrópica [35, 36].

No primeiro estágio que corresponde ao mapeador do esquema de compressão, calcula-se a transformada *wavelet* bidimensional da imagem. A transformada deve concentrar a maior parte da energia em uma pequena porção dos coeficientes. Essa característica permite que muitos coeficientes sejam eliminados, obtendo-se assim uma primeira compressão. Além disso, a forte relação da transformada *wavelet* com o mecanismo de visão humana permite que as componentes da imagem de menor importância visual sejam identificadas e eliminadas facilmente [36].

Como já foi enfatizado, a família de *wavelets* utilizada no processamento de imagens e especificamente em compressão de imagens, deve apresentar certas características como filtros de comprimento finito e fase linear, além de suavidade e regularidade. Ao se escolher uma família de *wavelets* deve-se levar em conta todos esses fatores [24].

Outro parâmetro, que pode interferir no desempenho do esquema de compressão é o número de camadas de decomposição/síntese. A taxa de compressão aumenta com o

número de camadas, embora esse aumento comece a saturar quando o número de camadas excede 4 [36]. Por outro lado, ao se aumentar o número de camadas, aumenta-se também a complexidade do sistema. Como conseqüência desse compromisso entre a complexidade do sistema e o ganho em taxa de compressão, geralmente são utilizadas de duas a quatro camadas.

No segundo estágio, os coeficientes da transformada *wavelet* são quantizados. Nesse estágio, distorções são introduzidas na imagem. Como cada subbanda da transformada *wavelet* possui características próprias, melhores resultados podem ser obtidos se cada subbanda for quantizada de forma independente [37].

Os tipos de quantizadores lineares mais utilizados são:

- **Quantizador com vetor de quantização pré-estabelecido**: é o mais simples de todos. A sua construção baseia-se na escolha de um passo diferente para cada subbanda. Cria-se um vetor q, denominado vetor de quantização, cujos elementos, $q(i)$, têm valores diretamente proporcionais aos passos de quantização utilizados em cada subbanda.
- **Quantização com alocação de bits:** esse quantizador utiliza a estatística das subbandas como base para o processo de quantização. Para cada subbanda, a transformada *wavelet* aloca uma certa quantidade de bits para sua representação, de acordo com as suas características estatísticas e a taxa de bits total que se deseja obter. O processo de alocação ótima de bits é realizado de acordo com o critério de minimização previamente escolhido [36, 38].
- **Quantização com Ponderação das Subbandas:** esse tipo de quantizador é um aperfeiçoamento do anterior. A ponderação do ruído de quantização de cada subbanda é realizada de acordo com a sensibilidade do sistema visual humano [36, 38].

No terceiro estágio, os coeficientes da transformada, já quantizados, são codificados entropicamente. Entre os códigos existentes, os mais utilizados são o código de Huffman (*run-length*) e o código aritmético. As subbandas de detalhes (médias e altas freqüências) são codificadas usando-se *run-length*. Já as subbandas de baixas freqüências não apresentam uma quantidade significativa de zeros que justifique a utilização desse código. Um código de Huffman tradicional é utilizado nesse caso.

3.2. COMPRESSÃO DE IMAGENS UTILIZANDO A TRANSFORMADA *WAVELET*

I. INTRODUÇÃO

Nos últimos anos houve um avanço significativo, no campo da compressão de imagens, utilizando-se a transformada *wavelet* [24, 39, 40]. O sucesso é principalmente atribuído às estratégias inovadoras para a organização e representação de dados de uma imagem transformada. Tais estratégias exploram de forma implícita ou explicita as propriedades estáticas dos coeficientes transformados em uma pirâmide *wavelet*. A maioria dos codificadores publicados, recentemente na literatura utiliza o algoritmo de decomposição piramidal (diádica), fundamentado no Cap.2. Um exemplo de tal decomposição é mostrado na Fig.3.2.

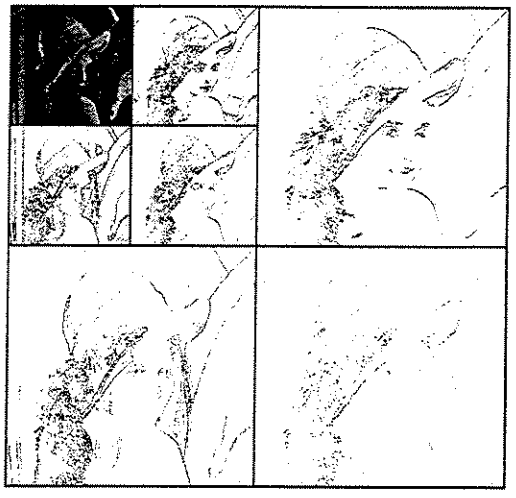


Fig. 3.2: Decomposição *wavelet* em duas escalas para a imagem Lena.

No presente, pode-se selecionar três algoritmos que representam bem os melhores da lista de codificadores *wavelets* para imagens. São eles: "*Embedded Zerotree Wavelet Coder*" (EZW) de Shapiro [40], "*Morphological Representation of Wavelet Data*" (MRWD) de Servetto & Cols [41] e "*Set Partitioning in Hierarchical Trees*" (SPHT) de Said & Perlman [42]. Ambos, o EZW e o SPIHT exploram a dependência entre subbandas de coeficientes *wavelets* pouco significativos, enquanto o MRWD realiza dentro de uma subbanda um tipo de *clusterização* de coeficientes *wavelets* significativos. Tais codificadores quando são comparados com o JPEG ("*Joint Photographic Experts Group*"), apresentam um

desempenho superior com relação a qualidade visual devido principalmente a eliminação do efeito de blocos. Quanto ao desempenho objetivo, medido a partir da PSNR ("*Peak Signal to Noise Ratio*") de imagens reconstruídas, tais algoritmos apresentam um aumento consistente da ordem de 1-3 dB sobre os codificadores por transformada de blocos.

Outro algoritmo, que segue a mesma linha do SPIHT e MRWD, é conhecido por "*Significance-Linked Connected Component Analysis*" (SLCCA) de Chais e Cols [43]. O SLCCA é uma extensão do MRWD, visto que além de explorar a *clusterização* de coeficientes significativos (a partir da análise de componentes conectados), também explora a dependência de *clusters* significativos entre subbandas. A dependência entre subbandas é explorada a partir do estabelecimento de um *link* de significância entre um *cluster*-pai (subbanda de baixa freqüência) e um *cluster*-filho (subbanda de alta freqüência).

Um algoritmo híbrido *wavelet*-fractal, chamado de "*Wavelet–Fractal Coder*" (WFC), foi desenvolvido por Li e Kuo e publicado em junho de 1999 [44]. Esse algoritmo utiliza operadores fractais no domínio *wavelet* para produzir coeficientes de alta resolução a partir dos coeficientes de baixa resolução. O resíduo é então codificado com um "*Layered Zero Coder*" (LCZ) [45], que é um codificador *wavelet* de plano de bit baseado no EZW. Essa estratégia de codificação do resíduo não é aplicada a toda a imagem, mas apenas às partes da mesma que justifiquem o cabeçalho extra. Trata-se de um codificador *wavelet* assistido pela predição fractal.

II. Propriedades Estatísticas das Transformadas *Wavelets* de Imagens.

A exploração das propriedades estatísticas das transformadas *wavelets* de imagem tem-se mostrado ser cada vez mais importante na compressão de imagens. Entre essas propriedades estatísticas, cinco se destacam:

1. Localização espaço-freqüência.

Um coeficiente *wavelet* é dito ser espacialmente localizado se contem apenas características de um segmento local de uma imagem de entrada. Devido à própria forma de decomposição de uma imagem em subbandas com quase nenhuma sobreposição, cada subbanda é localizada em freqüência com conteúdos aproximadamente independentes, como foi visto no Cap. 2. Cada coeficiente *wavelet* representa informações de uma certa banda de freqüências em uma determinada localização espacial.

2. Compactação de energia

Uma imagem natural é tipicamente composta de uma grande porção de regiões homogêneas e regiões de textura. Tal imagem apresenta ainda uma pequena porção de bordas que incluem objetos de fronteiras perceptivelmente importantes. Regiões homogêneas apresentam a menor variação e consistem basicamente de componentes de baixas freqüências. Regiões de texturas têm variação moderada e consistem de uma mistura de componentes de baixas e altas freqüências. As bordas por sua vez apresentam grande variação e são compostas basicamente de componentes de altas freqüências. Por construção, tendo-se por base o algoritmo piramidal, cada vez que uma subbanda passa-baixas em uma resolução fina é decomposta em quatros subbandas numa resolução mais grosseira, um processo de amostragem crítico permite gerar uma nova subbanda com apenas um quarto do tamanho da subbanda passa-baixas original. Repetindo-se esse processo de decomposição em uma imagem, isso resultará em uma compactação efetiva de energia em poucos coeficientes *wavelets*. Isso é devido a que grande parte da energia contida em regiões homogêneas e de textura, tende naturalmente a se concentrar numa subbanda passa-baixas.

3. *Clusterização* de coeficientes significativos em uma subbanda

Um coeficiente *wavelet c* é dito ser significante com relação a um determinado limiar T se $|c| \geq T$; caso contrário tal coeficiente é considerado insignificante. Um coeficiente insignificante também é conhecido como coeficiente zero. Devido à ausência de componentes de altas freqüências em regiões homogêneas e à presença de componentes de altas freqüências em regiões de textura e em torno das bordas, coeficientes significantes em subbandas passa-altas indicam usualmente a ocorrência de bordas e texturas com alta energia. Isso significa que tais coeficientes indicam uma proeminente descontinuidade ou uma proeminente mudança, um fenômeno que tende a ser *clusterizado*. A *clusterização* dentro da imagem Lena é mostrada na Fig. 3.3, sendo que o mapa de sementes para geração de cada um dos *clusters* é mostrado na Fig. 3.4.

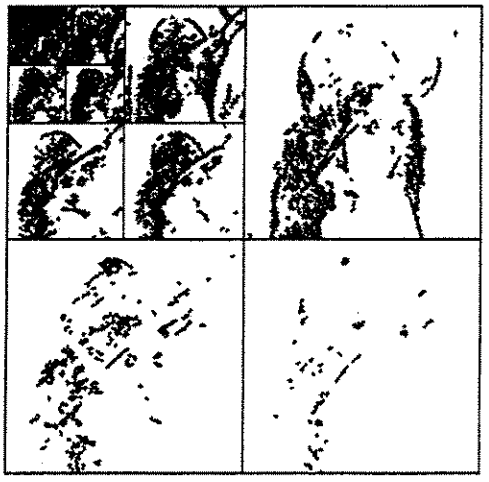


Fig.3.3: *Clusterização* de coeficientes significativos.

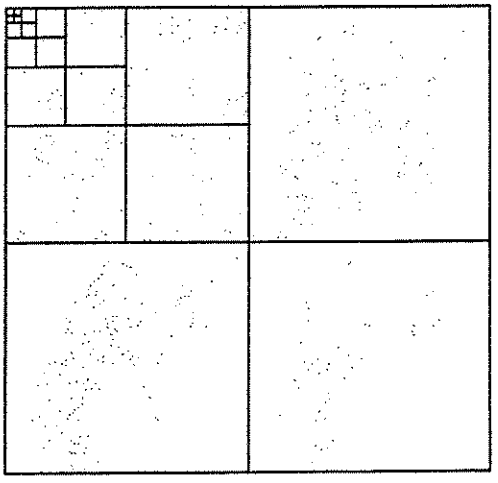


Fig. 3.4: Mapa de Sementes para a geração dos *clusters* significativos. A forma dos *clusters* é irregular e depende da semente que a origina.

4. Similaridade entre subbandas em diferentes escalas

Dado um coeficiente *wavelet*, como mostrado na Fig. 3.5, todos os coeficientes numa escala mais fina de orientação similar que correspondem a mesma orientação espacial são chamados seus descendentes, sendo por essa razão, o dado coeficiente chamado de ancestral. Especificamente, um coeficiente em uma escala grosseira é chamado de pai e todos os quatros coeficientes que correspondem à mesma localização espacial na próxima escala mais fina, são chamados de filhos. Embora a correlação entre os valores dos coeficientes pais e filhos seja extremamente pequena, há uma dependência adicional entre as magnitudes de ambos. Experimentos mostram que a correlação entre a magnitude

quadrática de um filho e a magnitude quadrática do pai tende a estar entre 0,2 e 0,6 com uma forte concentração em torno de 0,35. Essa dependência em magnitude quadrática caracteriza a similaridade entre as subbandas. Para mais detalhes, pode-se ver Shapiro [35, 40].

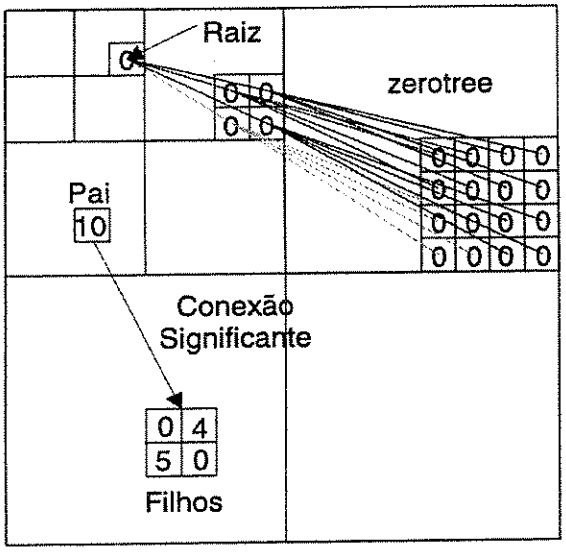


Fig. 3.5: Similaridade entre subbandas cruzadas de mesma orientação. A similaridade entre os coeficientes insignificantes (zeros) pode ser compactamente representada numa árvore de zeros (*zerotree*).

5. Decaimento da magnitude de coeficientes *wavelets* entre subbandas à medida que se refina a escala de resolução.

Embora pareça difícil caracterizar e utilizar por completo essa similaridade de magnitude entre descendentes, uma conjectura razoável baseada na experiência com imagens naturais é que a magnitude do filho é menor do que a magnitude do pai. Assumindo um campo de Markov aleatório ("*Markov random field*") como modelo da imagem, pode-se comprovar que estatisticamente a magnitude dos coeficientes *wavelets* decai de pai para filho. Mais precisamente, se a magnitude do coeficiente for medida por sua variância, pode-se mostrar que um pai tem uma variância maior do que a do filho.

III. Estratégias para a Organização e Representação de Dados

Existem duas propostas diferentes para se organizar e representar coeficientes *wavelets* na literatura [40, 41, 42, 43].

Primeira Abordagem: O EZW e o SPHT usam uma estrutura regular em árvore ou uma estrutura de conjuntos particionados em árvore para aproximar campos insignificantes ao longo de subbandas descendentes. Esses algoritmos exploram as propriedades estatísticas 1, 2, 4, e 5 de modo que se for encontrado um coeficiente pai zero, a probabilidade de seus descendentes serem zero é muito alta, principalmente a partir dos netos.

Segunda Abordagem: Adotada pelo MRWD, utiliza as propriedades estatísticas 1, 2, 3, e 4 para encontrar *clusters* irregulares de campos significativos dentro de subbandas.

Segundo a teoria de codificação de fonte, é amplamente aceito que em geral uma técnica de compressão de imagens torne-se cada vez mais computacionalmente complexa à medida que a mesma se torna mais eficiente. EZW interrompe essa tendência alcançando desempenho excelente com complexidade computacional muito baixa. Esse algoritmo identifica e aproxima regiões arbitrárias de zeros através das subbandas pela união de regiões altamente restritas à estrutura de árvores chamadas *zerotree*. Ao mesmo tempo, definem-se campos significativos em qualquer lugar fora dessas regiões de zero por meio de um refinamento progressivo das magnitudes dos coeficientes. Torna-se óbvio que cada *zerotree* pode ser efetivamente representado por seu símbolo raiz, como é mostrado na Fig. 3.5. Por outro lado, pode ainda haver muitos coeficientes zeros que não puderam ser incluídos na altamente estruturada árvore de zeros (*zerotrees*). A representação desses zeros isolados continua sendo muito cara para o codec.

O SPIHT procura enriquecer o EZW particionando a estrutura da árvore em três partes. São elas: a raiz da árvore, os descendentes filhos e os descendentes não filhos (netos, tataranetos, etc.). A última parte, os descendentes não filhos compreendem a maior parte da população na estrutura em árvore. Devido ao particionamento do *zerotree* pelo EZW, quando o mesmo encontra um coeficiente significativo o EZW é obrigado a codificar os outros três coeficientes filhos separadamente mesmo se os mesmos não forem significativos, impedindo assim uma maior ramificação da raiz. Por outro lado, o SPIHT trata todos os descendentes não filhos insignificantes como um conjunto e emprega um único símbolo (a terceira parte da estrutura) para representá-lo e codificá-lo. Nesse caso, se aparecer um filho significativo, a ramificação da árvore não será interrompida. Apesar dos dois últimos símbolos extras na representação da árvore, esse simples, porém refinado particionamento conduz a um significante aumento na PSNR de 0,86-0,94dB sobre o EZW

na imagem Lena. Isso indica que o SPIHT explora mais eficientemente a similaridade entre as subbandas que o EZW.

Diferentemente do EZW e do SPIHT, o MRWD atua diretamente nos coeficientes significativos, formando *clusters* com formas irregulares de coeficientes significativos dentro das subbandas como é mostrado na Fig. 3.4. Os *clusters* dentro de cada subbanda são progressivamente delineados por um contorno de zeros a partir de um operador morfológico de dilatação. Esse operador utiliza um elemento estruturante que controla a forma e o tamanho dos *clusters* assim como a formação de contornos. O MRWD gera então uma semente para cada *cluster*, como é mostrado na Fig. 3.4. Uma semente corresponde a um *pixel* no mapa de sementes do qual um *cluster* é originado. Em seguida, durante o processo de codificação esse *pixel* é especificado e a informação relativa ao seu posicionamento é codificada como *"overhead"*. Tendo em vista o grande número de *clusters* envolvidos, o total de cabeçalhos pode tomar uma fatia significante dos bits gastos na codificação da imagem. Apesar disso, o ganho em PSNR com relação ao EZW para a imagem Lena é da ordem de 0,78-0,95dB para diferentes taxas de distorção.

Em sua versão mais recente [41], o MRWD mantém sua característica principal, ou seja, a *clusterização* dentro de subbandas. Entretanto, o método de codificação foi modificado. A semente de cada *cluster* é especificada pela transmissão de um símbolo especial. Em adição, um modelo aritmético adaptativo baseado no contexto é empregado para a codificação em entropia, onde o contexto é baseado na significância dos coeficientes pais. O uso desse modelo aritmético adaptativo é uma exploração implícita da similaridade entre subbandas descendentes. Resultados experimentais mostram que essa última versão é superior à anterior.

Tal como o MRWD, o SLCCA explora a propriedade que os coeficientes *wavelets* significativos têm de tenderem a se agrupar em *clusters* irregulares (na forma) dentro de uma subbanda. A principal diferença entre esses algoritmos é que o SLCCA explora de forma mais efetiva a similaridade entre subbandas descendentes. Nesse contexto, o ponto chave chamado de conexão significante é o principal elemento novo inserido pelo SLCCA. Como foi dito anteriormente, a magnitude dos coeficientes *wavelets* decai estatisticamente de um pai para seus filhos. Isso significa que se um *cluster* foi formado numa subbanda de detalhes, então existe pelo menos um filho significante cujo pai também é significante. Em

outras palavras, um filho significante provavelmente pode ter sua origem determinada pelo pai através dessa conexão significante entre pai e filho. Na prática, essa conexão significante substitui a informação da posição do *cluster* filho. De fato, se a posição de um pai significativo em um *cluster* é conhecida, a posição do *cluster* filho pode ser inferida a partir dessa conexão significante, como mostra a Fig. 3.5. Isso é feito porque a codificação da conexão significante custa muito menos que a codificação direta da posição.

IV. Organização em Planos de Bits e Codificação Aritmética Adaptativa.

O último passo da grande maioria dos algoritmos de compressão de imagens envolve codificação entrópica. Em contraste com o modelo fixo, que trabalha bem fontes de Markov estacionárias, um codificador aritmético adaptativo atualiza a correspondente estimação de probabilidade condicional cada vez que o codificador percorre um contexto em particular. Para um fluxo de dados gerados por uma fonte não estacionária como uma imagem natural, a probabilidade condicional ou distribuição de probabilidades local pode variar bastante de uma seção para outra. O conhecimento da distribuição de probabilidade local, adquirida por um modelo adaptativo, é mais robusto do que uma estimativa global e consegue seguir as variações locais estatísticas de forma satisfatória. Em comparação com o modelo fixo, o codificador aritmético adaptativo é desse modo capaz de atingir maiores taxas de compressão.

Dada uma fonte de Markov não estacionária, é preferível organizar as saídas dessa fonte em um fluxo de dados de modo que cada distribuição local de probabilidades dê preferência a um símbolo da fonte. Essa é a idéia básica por trás da bem conhecida codificação sem perdas em um plano de bits, na qual sua imagem é dividida em planos de bits cada qual sendo codificado separadamente. As técnicas de codificação entrópicas podem então se tornar mais eficientes tendo em vista que planos de bits mais significativos contêm mais áreas uniformes.

Tanto o EZW quanto o SPIHT utilizam a codificação em planos de bits. O EZW, por exemplo, utiliza tal codificação através de uma transmissão progressiva de magnitudes, onde os "0" bits antes do primeiro "1" bit são codificados como *zerotree* ou zero isolado.

O SLCCA e o MRWD utilizam a codificação em plano de bits de modo a explorar toda a capacidade de um codificador aritmético adaptativo. A idéia é que dentro de uma subbanda

existem mais coeficientes com pequenas amplitudes do que aqueles com grandes amplitudes. Isso implica que os planos de bits mais significativos conteriam um número bem maior de zeros do que de uns. Dessa forma, o codificador aritmético adaptativo geraria distribuições locais de probabilidades mais acuradas na qual as probabilidades condicionais para símbolos "0" são aproximadamente 1(um) para os planos de bits mais significantes.

V. Avaliação do Desempenho dos Algoritmos.

Os algoritmos EZW, SPIHT, SLCCA, MRWD e WFC são comparados a seguir, dentro de certas condições.

Condições

- São utilizadas as imagens Lena (Fig.3.5), Bárbara (Fig.3.6), e Baboon (Fig.3.7). Todas elas em tons de cinza com dimensão 512 x 512
- Todas as imagens foram decompostas a partir de uma transformada *wavelet* diádica (pirâmide de subbandas), utilizando-se filtros *spline* biortogonais 9/7.
- Não foi utilizada alocação de bits ótima.
- Primeiro, os coeficientes *wavelets* foram quantizados com o mesmo quantizador escalar uniforme. Os resultados foram então codificados em planos de bits de acordo com o método determinado pelo algoritmo em questão.
- O desempenho desses algoritmos é medido da forma usual a partir de curvas de taxas de distorção, onde a taxa é medida em bits por *pixel* e a distorção PSNR é definida como,

$$PSNR(db) = 20\log_{10}\frac{255}{RMSE} \tag{3.1}$$

onde *RMSE* é a raiz quadrada do erro quadrático médio entre a imagem reconstruída e a imagem original.

Fig.3.5 Imagem Lena

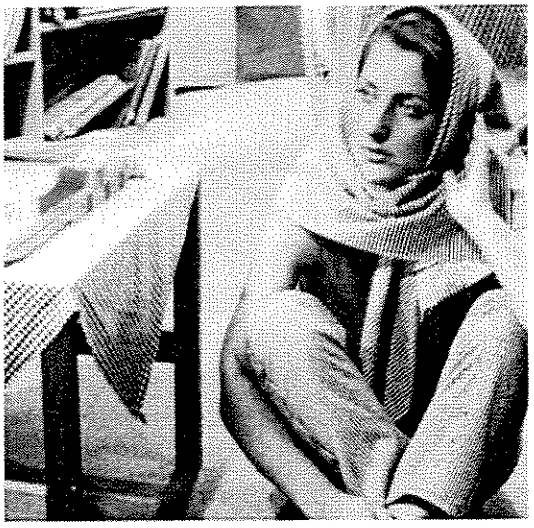


Fig.3.6. Imagem Bárbara

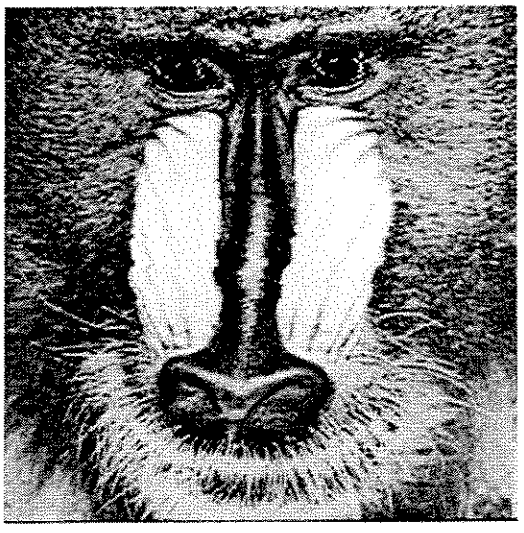


Fig.3.7 Imagem Baboon

Na Fig. 3.8, compara-se os desempenhos dos algoritmos EZW, SPIHT, MRWD, SLCCA e WFC a partir de suas curvas de taxas de distorção obtidas para a imagem Lena. Nessa mesma figura, apresenta-se como referência, o desempenho do JPEG para essa mesma imagem. Observa-se a partir desse resultado que os algoritmos *wavelets* para compressão de imagens, representados pelos algoritmos SPIHT, MRWD, SLCCA e WFC, têm um desempenho superior ao JPEG. Isso se torna mais evidente quando os resultados são comparados, tendo-se em vista que tais algoritmos não apresentam artefatos devido ao efeito de blocos tão comuns no JPEG. Entre os demais algoritmos, o EZW foi o que

apresentou pior desempenho. Isso era esperado porque o EZW é um referencial, devido a que foi o primeiro algoritmo que levou as *wavelets* ao topo do reconhecimento dentro dos compressores de imagem tanto em eficiência quanto em simplicidade (computacional). De fato, muitos algoritmos, entre eles o LZC de Taubman [45], foram desenvolvidos a partir do EZW. Nesse contexto, o algoritmo que mais se destacou como uma evolução significativa do EZW foi o SPIHT, apresentando alto desempenho em termos de taxa-distorção e simplicidade computacional. Pelos resultados da Fig.3.8, o MRWD e SPIHT tiveram desempenhos semelhantes a partir de conceitos diferentes. O MRWD explora as propriedades que os coeficientes significativos dentro de uma subbanda de detalhes apresentam, ou seja, tendem a se *clusterizar*, e o SPIHT explora a propriedade de dependência entre as subbandas de coeficientes insignificantes (zeros). O SLCCA obteve um desempenho superior ao MRWD e SPIHT. Resultado esse esperado, visto que o SLCCA explora tanto a *clusterização* de coeficientes como a dependência entre subbandas, porém de coeficientes significativos. O WFC obteve desempenho médio comparável ao do SLCCA para a imagem Lena.

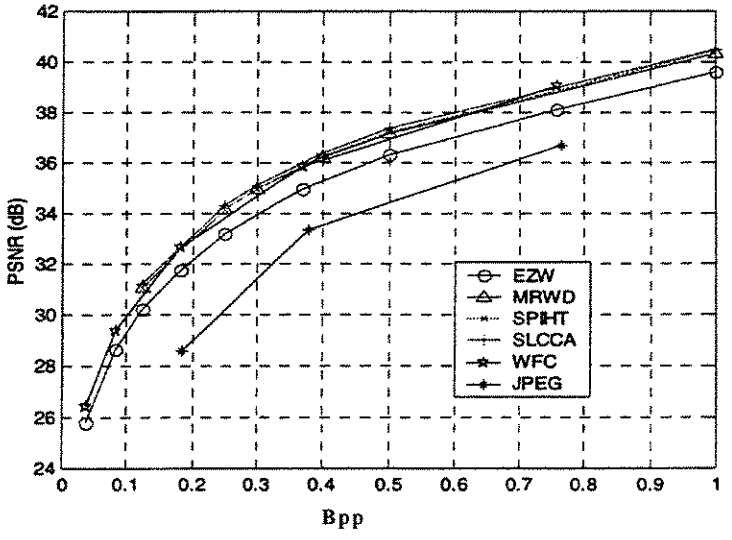


Fig.3.8: Desempenho do EZW, SPIHT, MRWD, SLCCA, WFC e JPEG para imagem Lena

Na Fig. 3.9, o desempenho dos algoritmos foi semelhante ao obtido para a imagem Lena. A diferença está no desempenho do WFC que para essa imagem foi ligeiramente superior ao SLCCA. Uma explicação para esse fenômeno está nas diferenças entre as imagens Lena e Barbara. A imagem Barbara apresenta um número bem maior de regiões de textura do

que a imagem Lena. Tanto o SLCCA quanto o WFC apresentam um bom desempenho em regiões de textura. Porém Lim [46] demonstrou que são exatamente as regiões de textura e de bordas as de predição fractal mais eficiente. Por esse motivo, o WFC teve um desempenho superior ao SLCCA para todas as taxas de bits analisadas para a imagem Barbara.

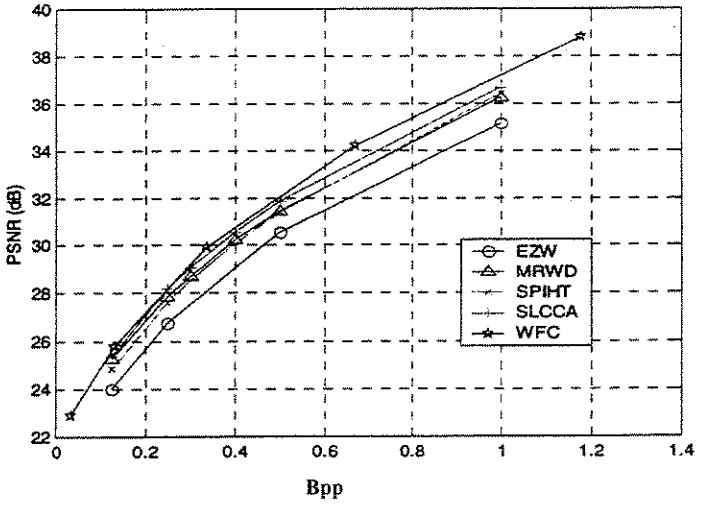


Fig.3.9: Desempenho do EZW, SPIHT, MRWD, SLCCA, e WFC para a imagem Bárbara .

Entretanto o desempenho mostrado na Fig.3.10 para a imagem Baboon mostrou uma inversão de desempenho. Nesse caso, o WFC obteve o pior resultado entre os codificadores avaliados mesmo sendo a imagem Baboon uma imagem com alto conteúdo de bordas. A explicação nesse caso é que o WFC utiliza a predição fractal em conjunto com o LZC [45] e tanto o LZC quanto o EZW não têm um desempenho tão elevado em imagens muito ricas em detalhes. Nesse caso, o codificador *wavelet* utilizado, diminuiu o desempenho do WFC.

Sugere-se então para solucionar esse problema a implementação da predição fractal com o SLCCA ou MRWD.

Salienta-se que apesar do SPIHT não ter apresentado o melhor desempenho, ele é em conjunto com o EZW, o algoritmo mais eficiente no aspecto computacional. Entre os algoritmos apresentados o que apresenta maior esforço computacional é o WFC, cuja técnica de predição fractal envolve intrinsecamente um processo de busca e otimização.

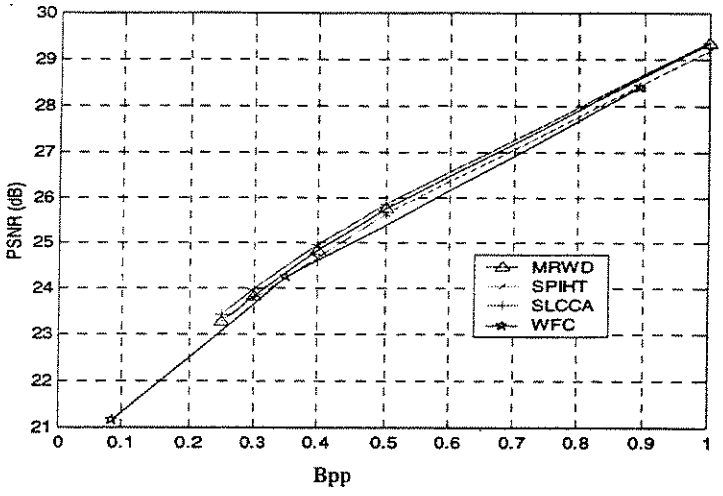


Fig.3.9: Desempenho do MRWD, SPIHT, SLCCA e WFC para a imagem Baboon.

3.3. A Transformada *Wavelet* na Compressão de Vídeo.

I. Introdução.

No cenário atual das telecomunicações, a compressão digital de vídeo, é uma área de grande importância. A maturidade alcançada pelas técnicas de compressão, a disponibilidade de tecnologia para se implementar codificadores a custos razoáveis conjuntamente com a crescente demanda pelos novos serviços de vídeo, tais como a HDTV (*"High Definition Television"*) têm aumentado o interesse por algoritmos que sejam eficientes e que satisfaçam as novas exigências dos serviços de telecomunicações.

A utilização da transformada *wavelet* nessa aplicação é uma conseqüência dos bons resultados na compressão de imagem estáticas. Como foi visto na seção anterior existem na literatura codificadores que usando essa transformada apresentam resultados muito satisfatórios [12, 42, 47, 48]. Enquanto o problema de compressão de vídeo for encarado de uma forma diferente daquela em que simplesmente se codifica diretamente uma imagem, principalmente devido a presença de movimento, é de se esperar que prósperas técnicas de compressão de imagens, freqüentemente farão parte dos algoritmos de codificação de vídeo [12].

II. FUNDAMENTOS DA COMPRESSÃO DE VÍDEO

O objetivo do codificador de vídeo é a redução da taxa de bits necessária tanto para o armazenamento quanto para a transmissão. Isso é feito através da exploração de redundâncias de maneira que apenas um conjunto mínimo de informações seja codificado entropicamente. O desempenho da compressão, entretanto depende das técnicas utilizadas e do conteúdo das cenas. Em sistemas práticos, um compromisso entre o desempenho e a complexidade deve ser levado em conta.

O processo de compressão é realizável devido a que o número de bits necessários para se representar uma imagem pode ser reduzido, explorando-se a redundância natural das seqüências de imagens. Em geral pode-se identificar os seguintes tipos de redundância:

Redundância psico-visual: O olho humano não reage com igual intensidade a todas as informações visuais contidas em uma imagem. Algumas informações têm menor importância relativa, ou seja, são psico-visualmente redundantes, podendo ser eliminadas sem que uma perda na qualidade da imagem possa ser percebida. Essa redundância é explorada através da subamostragem da crominância.

Redundâncias espacial e temporal: As seqüências de vídeo possuem esses tipos de redundâncias [49, 50, 51]. As correlações exploradas pelas técnicas de compressão são especialmente aquelas entre *pixels* de um mesmo quadro (espacial) ou entre *pixels* de quadros adjacentes (temporal), ou seja, assume-se que a magnitude de um *pixel* de uma imagem pode ser predito a partir dos *pixels* vizinhos do mesmo quadro (utilizando-se a codificação intraquadros) ou de *pixels* de um quadro adjacente (utilizando-se a codificação interquadros).

A codificação intraquadros é a maneira mais simples, tanto conceitualmente quanto em termos de implementação, de se obter uma compressão de seqüências de vídeos. Nessa abordagem cada quadro é tratado como uma imagem estática sendo que uma técnica de compressão de imagem é então aplicada [51]. A codificação interquadros é uma técnica mais elaborada do que a anterior. Explora a dependência entre os quadros consecutivos do sinal de vídeo, ou seja, a redundância temporal da seqüência. Para isso é utilizada uma técnica denominada predição com compensação de movimentos [49, 50] a qual se associa a codificação DPCM ("*Differential Pulse Code Modulation*").

A compensação de movimentos é um conceito que está baseado na estimação de movimentos entre quadros. Se o processador de redução de redundância temporal emprega compensação de movimento, sua saída pode ser expressa do seguinte modo:

$$e(x, y, t) = I(x, y, t) - I(x - u, y - v, t - 1) \tag{3.2}$$

onde $I(x, y, t)$ são os valores dos *pixels* na localização espacial (x, y) no $quadro(t)$ e $I(x - u, y - v, t - 1)$ são os correspondentes valores dos *pixels* na localização espacial $(x - u, y - v, t)$ no $quadro(t - 1)$. As coordenadas (u, v) de saída do estimador de movimento, define o movimento relativo de um bloco de um quadro para outro e é chamado de vetor de movimento do bloco centrado em (x,y).

Resumindo, se uma cena apresenta uma quantidade de movimento relativamente baixa, a melhor predição é freqüentemente obtida a partir dos *pixels* do quadro adjacente temporalmente. O movimento dos *pixels* pode ser descrito por um número limitado de parâmetros de movimentos, isto é, os vetores que descrevem a trajetória de deslocamento dos *pixels* entre quadros consecutivos.

A estimação de movimentos é o processo de determinação dos vetores de movimento. Os vetores resultantes são transmitidos para o receptor. Nesse, o decodificador identifica qual é a área da imagem de referência utilizada para a predição através de uma compensação de movimento. Soma-se então, essa predição com a diferença decodificada para se obter uma imagem reconstruída. A compensação de movimento também é realizada no codificador para se calcular o erro de predição que será codificado [12, 52, 53, 54, 55, 56, 57, 58].

III. Codificação com Escalonabilidade.

III.1 Introdução

A codificação com escalonabilidade consiste, na codificação do sinal em vários níveis de hierarquia sendo que a decodificação em determinado nível dessa hierarquia depende da disponibilidade de dados de todos os níveis inferiores a esse nível.

Para se obter sinais de vídeo com diferentes resoluções (espaciais, temporais) ou níveis de qualidade, são utilizados sistemas escalonáveis, também chamados de codificadores em camadas, ou codificadores hierárquicos. O objetivo desses sistemas é o de oferecer uma compatibilidade ou interoperabilidade entre os diferentes serviços ou padrões. Por exemplo,

para a transmissão dos sinais de televisão digital e de HDTV ("*Hight Definition TV*"), a compatibilidade entre os vários sistemas é vital. É importante e necessário que o usuário de TV convencional possa receber o sinal de HDTV e obter um sinal de menor resolução (correspondente a TV convencional) sem ter que decodificar todo o sinal recebido [49, 50, 51].

Existem basicamente duas formas de se obter seqüências de vídeos com escalonabilidade: a codificação *simulcast* e a codificação com escalonabilidade.

III.2. CODIFICAÇÃO *SIMULCAST*

Na codificação *simulcast* cada camada de vídeo corresponde a um tipo de resolução ou qualidade e é codificada independente das outras. A banda de freqüência disponível é particionada de acordo com a resolução ou qualidade que se deseja atribuir a cada camada. Nessa codificação são utilizados decodificadores independentes para cada camada.

III.3. CODIFICAÇÃO COM ESCALONABILIDADE.

Na codificação de vídeo com escalonabilidade, a primeira camada de vídeo é codificada independentemente, enquanto as demais são codificadas de acordo com a anterior. A primeira camada é denominada camada base e transporta um sinal com qualidade ou resolução básica. Essa camada, pelo fato de ter sido codificada independentemente, pode ser decodificada por decodificadores não-escalonáveis. As outras camadas, denominadas camadas de enriquecimento transportam as informações de refinamento ou enriquecimento e quando são adicionadas ou combinadas com a camada da base, geram um sinal com qualidade ou resolução mais alta.

Geralmente a codificação com escalonabilidade é mais eficiente do que a *simulcast*, uma vez que com exceção da camada base, todas as outras camadas são codificadas utilizando-se parte da banda de informação atribuída à camada anterior. O aumento da eficiência, entretanto vai depender do tipo de técnica utilizada, do número de camadas e da partição da banda de freqüências. Esse aumento implica, porém em um aumento da complexidade em relação à codificação *simulcast*. Em geral, utiliza-se um codificador com escalonabilidade com duas camadas: uma camada de base e uma camada de enriquecimento.

III.4. CODIFICADORES COM ESCALONABILIDADE VERSUS CODIFICADORES SEM ESCALONABILIDADE.

1. A codificação em camadas possui um poder de recuperação contra erros na perda das camadas, pois garante que os dados contidos na camada básica sejam recuperados garantindo dessa forma uma qualidade mínima, aumentando a robustez do sistema em canais sujeitos a erros de transmissão. Isso é importante, por exemplo, em redes do tipo ATM ("*Assynchronous Transfer Mode*") como base para o sistema B-ISDN ("*Broadband Integrated Servise Digital Network*") e em aplicações de radiodifusão através de satélites ou distribuição terrestre. Apesar de ser possível empregar códigos corretores de erro para se reduzir os erros de transmissão é mais interessante aceitar que alguns erros cheguem ao receptor de vídeo e tentar disfarçá-los. Os códigos corretores devido a seu alto custo podem ser utilizados na proteção apenas dos dados mais sensíveis.
2. A maior vantagem da codificação em camadas e a sua capacidade de oferecer compatibilidade entre os padrões recomendados pelo CCITT ("*Comitté Consultatif International Telegrafique et Telephonique*" – atualmente ITU-T-"*International Telecommunications Union*"), de codecs para ISDN (banda estreita) e para B-ISDN (banda larga). A camada de maior prioridade do novo codificador B-ISDN deve ser idêntica a do sinal completo do codificador ISDN. Assim, os sinais emitidos pelos decodificadores mais novos podem ser decodificados pelos mais antigos sem um custo extra. As camadas de prioridade menores contêm versões finamente quantizadas da diferença entre o sinal completo e a primeira camada [59].

Existem cinco tipos básicos de escalonabilidade: Temporal, Espacial, de Dados, Relação Sinal-Ruído, e Híbrida. Nas seções seguintes eles são apresentados.

III.5. ESCALONABILIADADE TEMPORAL.

Consiste em se codificar duas camadas com resoluções temporais próprias, de maneira que quando as camadas são combinadas obtém-se uma resolução temporal completa igual a do sinal de entrada. A taxa de entrada de quadros é simplesmente particionada entre a camada de base e as camadas de enriquecimento. Esse tipo de escalonabilidade pode ser utilizado para se obter interoperabilidade entre o vídeo entrelaçado e o progressivo, por exemplo, em HDTV, para o qual a migração da primeira geração de HDTV entrelaçada

para a versão de HDTV progressiva de alta resolução temporal é possível de maneira compatível. Há também outras aplicações, tais como, a geração de novos serviços de comunicações utilizando-se resolução de TV com um formato progressivo com o dobro da taxa de quadros. Essa aplicação exige uma interoperabilidade com os *displays* de TV existentes.

III.6. ESCALONABILIDADE ESPACIAL

É uma codificação em camadas, onde cada camada possui uma resolução espacial própria, mas uma mesma resolução temporal e qualidade. As camadas são fracamente correlacionadas sendo que existe uma considerável liberdade na utilização de resoluções e formatos de vídeos em cada camada. A resolução espacial é incrementada quando se passa de determinada camada para a camada superior. A camada mais alta utiliza uma predição espacial intercamadas, com relação às imagens decodificadas da camada inferior. Essa predição é responsável pelo aumento do desempenho do codificador. Uma aplicação desse tipo de escalonabilidade é permitir a interoperabilidade, na transmissão dos sinais de TV digital e HDTV, em que a resolução da TV deve ser incrementada para atender os requisitos do sistema de HDTV.

III.7. PARTICIONAMENTO DE DADOS

Esta técnica consiste na partição de um sinal já codificado em duas camadas ou partições. No caso do sinal de vídeo a seqüência codificada é segmentada em duas ou mais camadas para transmissão ou armazenamento. A cada camada pode ser atribuída uma prioridade diferente o que possibilita, uma transmissão menos sensível a erros. Através desta técnica pode-se obter, um escalonamento por separação espectral. Esta técnica pode ser utilizada na transmissão de vídeo em redes ATM.

III.8. ESCALONABILIDADE SNR (RELAÇÃO SINAL- RUÍDO)

Neste caso o escalonamento gera camadas individuais de sinal codificado, com nível de qualidade diferente, mas com a mesma resolução espacial ou temporal. O escalonamento é obtido através da requantização dos coeficientes. A camada base transporta os coeficientes com a qualidade mais baixa, enquanto a camada de enriquecimento transporta os coeficientes de refinamento. Quando esses últimos coeficientes são adicionados à camada mais baixa enriquecem a relação sinal ruído (SNR). O que muda de uma camada para outra

é a qualidade de reprodução, devido a essa característica, se apenas a camada base for recuperada no receptor, a imagem reconstruída tem uma qualidade básica. Este tipo de escalonabilidade pode ser aplicado na transmissão de sinais de TV digital (ou HDTV) com deferentes níveis de qualidades.

III.9. Escalonabilidade Híbrida

Nesta técnica são combinadas dois ou mais tipos diferentes de escalonabilidade. Se duas técnicas de escalonabilidades são combinadas para se formar uma híbrida, então são obtidas três camadas (como mínimas): a camada base, a primeira camada de enriquecimento, e a segunda camada de enriquecimento. A primeira camada de enriquecimento é a camada imediatamente inferior à segunda camada de enriquecimento, da mesma forma que a camada base é a camada imediatamente inferior à primeira camada de enriquecimento.

3.4. A Escalonabilidade e a Transformada *Wavelet*

I. Introdução

A transformada *wavelet* provê uma maneira fácil de se implementar a codificação com escalonabilidade espacial. Isso se deve ao fato da decomposição *wavelet* apresentar a propriedade de escalonabilidade de forma natural, uma vez que, ao se decompor uma imagem utilizando-se a transformada *wavelet* obtem-se uma outra imagem com uma resolução espacial inferior e três imagens de enriquecimento.

II. Codificação de Vídeo Escalonável baseada na Transformada Wavelet

Na Fig. 3.10 apresenta-se uma proposta de um codificador de vídeo com escalonabilidade espacial baseado na transformada *wavelet* [60]. O sinal de entrada é decomposto pela transformada *wavelet*, produzindo quatro imagens resultantes diferentes: a subbanda de baixas freqüências e as subbandas de detalhes. A subbanda de baixas freqüências é codificada utilizando-se um codificador de vídeo com compensação de movimentos, que pode ser baseado na transformada *wavelet* ou na DCT. As imagens de detalhes passam por um estágio de compensação de movimentos. Em seguida são codificadas por um codificador EZW [40].

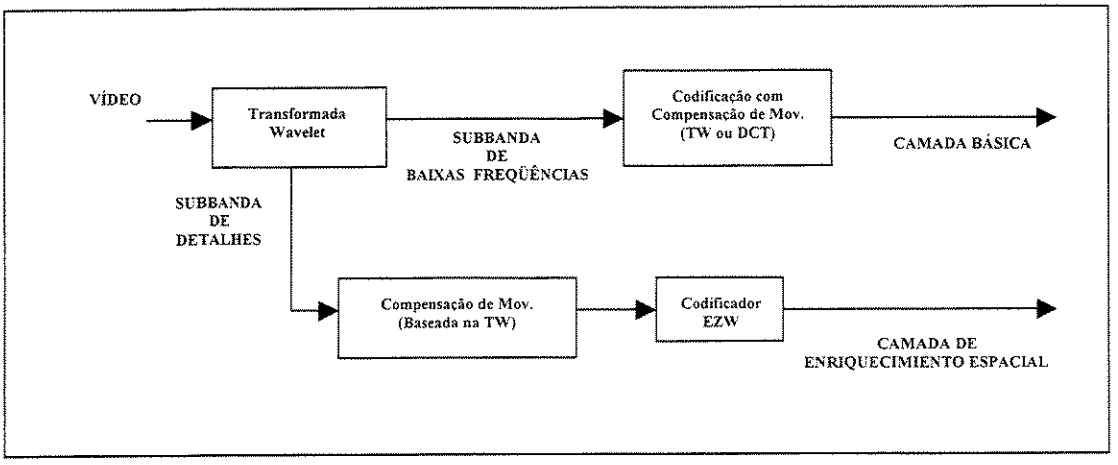


Fig. 3.10: Codificador com escanolabilidade espacial baseado na transformada wavelet.

Em geral, assim como no caso de imagens estáticas, podem ser utilizadas de 2 a 5 camadas de decomposição.

A redundância temporal de uma seqüência de vídeo é eliminada pela compensação de movimentos sendo que os quadros de erro de predição são então codificados e quantizados. Evidentemente existem outras formas de codificação além da apresentada na Fig. 3.10. Por exemplo, a compensação de movimentos pode ser realizada antes da TW ou ainda os quadros de erro de predição podem ser descorrelacionados utilizando-se uma DCT ("*Discrete Cosine Transform*").

Nessa estrutura, a escolha do algoritmo de estimação de movimento é crucial e determina o desempenho e complexidade do codificador [60]. Zhang et al [48] propuseram uma técnica de estimação de movimentos com multiresolução (MRME) que estima os vetores de movimentos no domínio da TW. Nessa técnica a estimação dos vetores de movimento é realizada de acordo com o índice da camada de decomposição. Os vetores de movimento das subbandas de detalhes e a subbanda de baixas freqüências da última camada de decomposição são calculados através de um algoritmo de estimação de movimento tradicional, semelhante ao utilizado pelo MPEG [49]. Os vetores de movimento das demais camadas são calculados através de uma predição que utiliza os vetores da próxima camada como referência.

Considere-se por exemplo, uma decomposição *wavelet* em três camadas. Os vetores das subbandas $D^1_{2^2}, D^2_{2^2}$ *e* $D^3_{2^2}$, da segunda camada de decomposição são preditos a partir dos vetores das subbandas $D^1_{2^3}, D^2_{2^3}$ *e* $D^3_{2^3}$, da terceira camada.

Assim como as técnicas tradicionais, a estimação de movimentos na MRME é realizada com base em blocos de *pixels*. Entretanto, os blocos utilizados nessa técnica possuem tamanhos diferentes de acordo com a camada de decomposição. Assim, por exemplo, para uma decomposição em 3 camadas utilizando-se um bloco n x n de referência, as subbandas da primeira, segunda e terceira camadas são divididas em blocos de tamanho n x n, $2n$ x $2n$, e $4n$ x $4n$. Dessa forma, garante-se que o número de blocos em todas as subbandas seja idêntico.

Em relação aos esquemas com escalonabilidade baseados na DCT, esse tipo de codificador segundo [60]apresenta melhores valores de SNR e uma qualidade subjetiva superior. A TW não introduz os efeitos de blocagem e reduz muito o ruído *mosquito* [42, 47, 48].

CAPÍTULO 4.

Subsistema Codificador de Imagem baseado na Transformada *Wavelet* para a Codificação dos Sinais de Vídeo

I. INTRODUÇÃO

No Capítulo 3 foi apresentado um estudo comparativo entre os algoritmos *"wavelets"*, considerados no estado da arte em compressão de imagens. Foram também mostrados, os princípios e conceitos utilizados por esses algoritmos considerados promissores em compressão de imagens. Concluiu-se que, nos últimos anos, o avanço significativo na codificação de imagens utilizando-se transformada *wavelet* (TW) deve-se principalmente às estratégias inovadoras para a organização e representação dos dados da imagem transformada. Destacou-se que a utilização da TW na compressão de sinais de vídeo é uma conseqüência dos bons resultados obtidos na compressão de imagens.

Vídeo digital consiste basicamente de uma seqüência de imagens. Dessa forma, técnicas de codificação de imagem são também utilizadas para codificação de vídeo. A principal diferença entre imagem e vídeo digital é que o vídeo tem uma dimensão adicional, a dimensão temporal. Técnicas adicionais podem ser utilizadas para se extrair a redundância associada a essa dimensão, por exemplo, a codificação de diferenças entre quadros usando-se algoritmos de estimação e compensação de movimento.

Nesta seção apresenta-se um codificador de imagem baseado em TW que foi implementado para ser utilizado como um dos blocos do codificador de vídeo digital.

II. IMPLEMENTAÇÃO DA TRANSFORMADA *WAVELET*

A análise de multiresolução bidimensional (MRA) [13] de uma imagem, $f(x,y)$, pode ser implementada através da utilização da transformada *wavelet*. Neste trabalho, assim como na maioria dos codificadores publicados recentemente na literatura, utiliza-se uma generalização do caso unidimensional para o caso bidimensional [12, 13, 18, 41, 44, 60, 61].

O algoritmo unidimensional é inicialmente aplicado às linhas da imagem e depois às colunas. A idéia é primeiramente realizar uma MRA para cada linha da imagem,

substituindo-se o valor original de cada pixel pelo coeficiente MRA e a seguir repete-se o processo para cada coluna da imagem. Nos Cap.2 e 3, já apresentados, pode-se obter os detalhes da fundamentação matemática desse procedimento.

Após o primeiro estágio de decomposição, são geradas 4 subbandas. A subbanda que contém os coeficientes de baixas freqüências é chamada LL_1, e pode ser considerada uma aproximação de baixa resolução da imagem original. As subbandas que contêm os coeficientes de alta freqüência na vertical, horizontal e diagonal, são chamadas LH_1, HL_1, HH_1, respectivamente.

O próximo estágio de decomposição segue acoplado à banda LL_1. Ele gera as subbandas LL_2, LH_2, HL_2, HH_2, onde analogamente LL_2 é uma aproximação de LL_1 em baixa resolução. Para N níveis de decomposição, serão geradas $3N$ subbandas de detalhamento (subbandas de alta freqüência). A Fig. 4.1 ilustra esse processo.

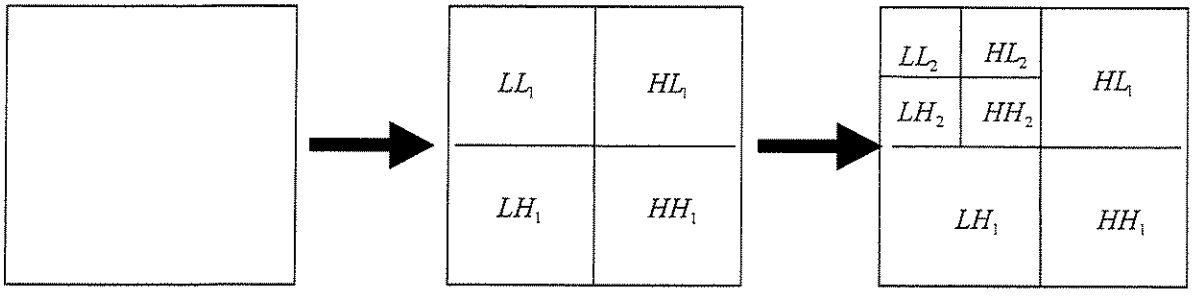


Fig. 4.1: Dois primeiros estágios de decomposição *wavelet* da imagem.

São também importantes as dimensões de cada subbanda geradas a partir da imagem original, de dimensões V linhas por H colunas. Cada estágio da decomposição em subbandas é implementado através da aplicação dos filtros às linhas e às colunas da aproximação obtida (LL_n para o *n-ésimo* estágio). Os filtros unidimensionais utilizados nessa decomposição são os filtros conjugados em quadratura $\tilde{H}$ e $\tilde{G}$ que possuem respostas impulsivas $h(-n)$ e $g(-n)$, respectivamente [13, 18]. Após essas convoluções, as linhas ou colunas da imagem são dizimadas por um fator de 2. No entanto, a convolução linear de um sinal discreto de um certo comprimento N_1 com um filtro qualquer de comprimento N_2 produz uma saída de comprimento $N_1 + N_2 - 1$ [10]. De acordo com o

esquema de decomposição MRA bidimensional, para conservar o número total de amostras, as convoluções devem manter o comprimento N_1 original das linhas e colunas. Dessa forma, após a dizimação, cada linha ou coluna terá um comprimento exatamente de $N_1/2$ onde N_1 é par. Para se obter esse resultado há duas alternativas [62]. Uma delas é a utilização da convolução circular e a outra é utilizar a extensão simétrica da linha ou coluna a ser filtrada. Ambas as alternativas exibem a mesma complexidade computacional e satisfazem o critério de reconstrução perfeita. A técnica de extensão simétrica foi utilizada neste trabalho devido às vantagens que oferece do ponto de vista de compactação de energia, conforme será visto em detalhes na próxima seção [62, 63].

II.1. Implementação da Convolução com Extensão Simétrica

Um dos problemas relativos ao uso de bancos de filtros para a codificação de dados ou sinais de duração finita, como vetores de linha ou colunas em uma imagem digital, é a manipulação de condições de contorno nas bordas do vetor. Torna-se necessário definir uma extensão y do vetor finito x, para computar os coeficientes da transformada quando os filtros se estendem além do vetor com o qual está convoluindo. As soluções de extensão periódica ou com zeros (*zero padding*) introduzem descontinuidades no sinal de entrada, x. Essas descontinuidades adicionam variância nas subbandas de detalhes ou altas freqüências reduzindo o ganho de codificação.

As transformadas que conservam o número de amostras entre o sinal discreto ou vetor original e sua representação no domínio da transformada são chamadas "não expansivas". A TW sendo implementada através de bancos de filtros mostra ser então uma transformada expansiva, já que as convoluções do sinal de entrada com os filtros aumentam o comprimento do sinal.

Seja N_0 o comprimento do vetor de entrada $x(n)$ em um estágio unidimensional de uma decomposição MRA, como mostram as figuras Fig.2.8 e Fig.2.9. Os filtros biortogonais de decomposição passa-baixas e passa-altas $h(-n)$ e $g(-n)$ possuem comprimentos p e q, respectivamente. A decomposição MRA terá como resultado dois vetores de comprimento $\frac{(N_0+p-1)}{2}$ e $\frac{(N_0+q-1)}{2}$ totalizando assim $N_0+\frac{(p+q-2)}{2}$ coeficientes. Caso o sinal de entrada seja uma imagem de dimensões $V \times H$, a decomposição MRA levaria a

$VH + V\frac{(p+q-2)}{2} + H\frac{(p+q-2)}{2}$ coeficientes. Para valores típicos de $V = H = 512$, $p = 9$ e $q = 7$, tem-se um aumento de aproximadamente 5,5% no número de amostras necessárias para se representar o sinal em somente um estágio de decomposição. Claramente esse é um problema que vai de encontro à finalidade primária de se comprimir o sinal.

Caso certas restrições relacionadas ao sinal de entrada e aos bancos de filtros sejam satisfeitas, a TW implementada na forma de bancos de filtros pode tornar-se uma transformada não expansiva [62, 63]. Brislawn verificou que se o comprimento N_0 da entrada unidimensional for par, e se os filtros exibirem simetria e possuírem comprimentos menores que N_0, então a MRA poderá ser implementada sem causar aumento no comprimento do vetor de entrada, através de uma convolução circular entre uma extensão simétrica do sinal de entrada e os filtros. Felizmente esse é o caso dos bancos de filtros biortogonais.

As descontinuidades introduzidas pela implementação circular são eliminadas porque se utiliza uma extensão par do sinal de entrada. Isso dobra o comprimento da entrada, aparentemente dobrando o número de coeficientes resultantes. A chave para se implementar esse procedimento de forma não expansiva está em se usar filtros de fase linear que produzam subbandas simétricas. Guardando metade de cada subbanda simétrica, o tamanho total da transformada é reduzido a N_0 coeficientes, tornando-a não expansiva. Para realizar esse procedimento deve-se determinar as combinações de extensões simétricas da entrada com as simetrias e fases dos filtros que geram as subbandas simétricas.

Os detalhes do procedimento e as derivações algébricas podem ser consultados em [62, 63].

Nas simulações deste trabalho foram utilizadas extensões simétricas do tipo $E_S^{(1,1)}x(n)$. Essa notação indica que o sinal de entrada $x(n)$, $n = 0,1,2...,N_0 - 1$ é estendido para o sinal $y(n)$, $n = 0,1,2,...,2N_0 - 3$, onde a primeira e a última amostra de $x(n)$ somente aparecem uma vez na extensão $y(n)$, conforme ilustrado na Fig. 4.2.

Na Fig. 4.2 observa-se que:

- Um segundo período de $y(n) = E_s^{(1,1)} x(n)$ é representado, para tornar visível que, essa técnica não introduz descontinuidade na convolução periódica ou circular (com período de $2N_0 - 2$) de $y(n)$ com os filtros biortogonais simétricos.
- $y(n)$ exibe simetria em torno de 0.
- Chamando α_1 a amostra de simetria (ou atraso em grupo) do filtro simétrico de decomposição, o resultado da convolução de $y(n)$ com o filtro será simétrico com relação a $n = \alpha_1$.
- A dizimação por 2 consiste em salvar as primeiras $N_0/2$ amostras alternadas a partir de $n = \alpha_1$.

No estágio de reconstrução, insere-se zeros entre essas $N_0/2$ amostras e realiza-se novamente a extensão simétrica do tipo $E_s^{(1,1)}$. A seguir desloca-se circularmente o vetor de α_1 unidades à direita e procede-se com a filtragem dessa vez com o filtro de síntese, de atraso em grupo α_2. O resultado dessa última convolução tem ainda comprimento $2N_0 - 2$ e simetria em torno de $n = \alpha_1 + \alpha_2$. O resultado final são as primeiras N_0 amostras a partir de $n = \alpha_1 + \alpha_2$. É importante ressaltar que esse esquema de convoluções com extensão simétrica também provê reconstrução perfeita.

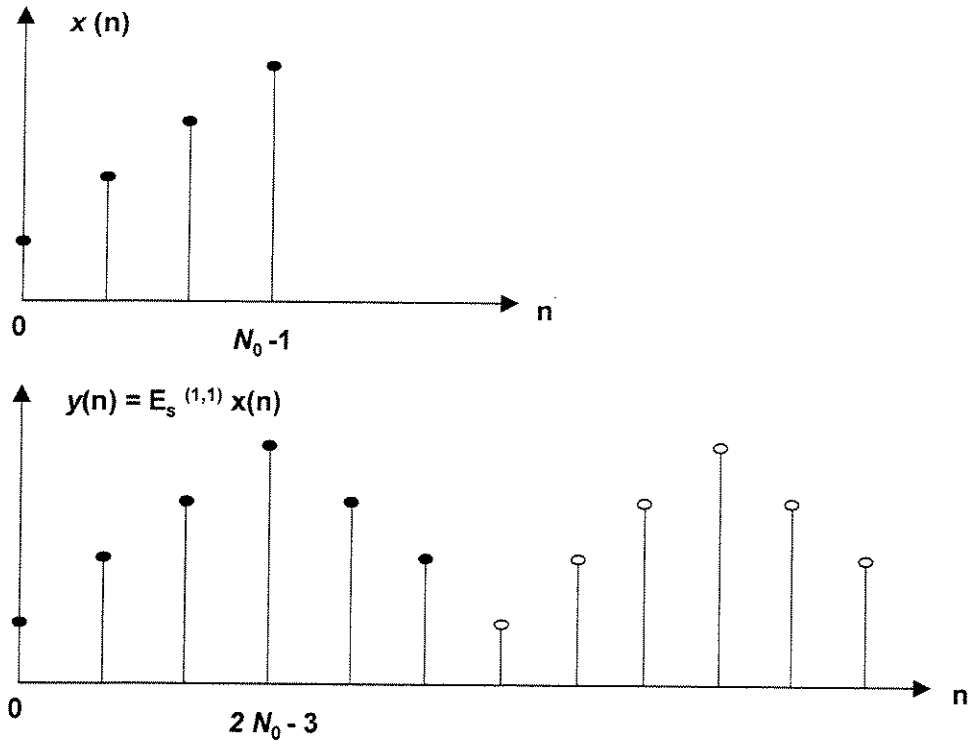


Fig. 4.2: Extensão simétrica par de um sinal $x(n)$.

III. Estrutura Regular em Árvore para a Organização e Representação dos Dados

Uma importante propriedade estatística, apresentada no Cap. 3, de similaridade entre subbandas em diferentes escalas e o decaimento da magnitude de coeficientes *wavelets* entre subbandas à medida que se refina a escala de resolução é mostrada nas figuras Fig.3.2 e Fig. 3.5.

Conforme apresentada no capítulo anterior, a idéia consiste em se definir uma árvore de símbolos nulos ou zeros (*zerotree*) que começa em uma raiz na decomposição diádica que também é nula. Algumas dessas raízes são exemplificadas na Fig. 3.5. Dessa forma, essa raiz pode ser nomeada como o fim de bloco relativo a toda a árvore de zeros. Como a árvore cresce em potências de 4, uma arvore de zeros permite descartar a representação de muitos símbolos insignificantes de uma só vez. Deve-se notar também que uma árvore de zeros compreende coeficientes que correspondem a uma mesma localização espacial na imagem original.

Recordando brevemente, foi citado que na implementação original do algoritmo *wavelet zerotree* de Lewis & Konwles (1992) [64] são nomeados zeros todos os descendentes do

coeficiente (raiz) quantizado como zero. Porém, quando um coeficiente é quantizado como zero, a probabilidade de que seus descendentes, na estrutura de subárvore *wavelet*, sejam também zero é muito alta. Mesmo assim, é possível a existência de algum descendente significativo (não zero). Essa possibilidade gera uma fonte de erro no algoritmo de Lewis & Konwles.

Na próxima seção, apresenta-se uma implementação da codificação *zerotree*. A implementação particular desenvolvida aqui está baseada nas idéias usadas por Lewis & Konwles (1992), Shapiro (1993), Said & Perlman (1996), descritas no Cap.3.

III.1. Implementação da Codificação *Wavelet* de Árvore de Zero

O algoritmo desenvolvido apresenta as vantagens da estrutura mostrada na Fig. 4.3. Esse algoritmo utiliza a técnica conhecida como "transmissão progressiva". Transmissão progressiva é um processo hierárquico pelo qual uma imagem é enviada por partes, ou camadas. Ou seja, uma versão de baixa resolução da imagem é enviada de maneira que se garanta uma qualidade básica. Os detalhes da imagem, ou informações de mais alta resolução, são enviados de uma maneira gradual. Essa é uma propriedade desejável em aplicações como a *Internet*, onde o usuário pode observar a imagem intermediária e decidir se espera por um *download* completo, ou se passa para algo mais interessante. O fato de o usuário poder interromper a qualquer instante a transmissão da imagem representa uma redução dos dados efetivamente transmitidos. Essa redução é chamada de compressão efetiva.

III.1.1. Terminologia e Definições

É necessário definir a terminologia utilizada para o tratamento da estrutura de árvore mostrada na Fig. 4.3. "Superior" refere-se ao topo da estrutura, perto dos coeficientes *wavelets* de mais baixa resolução. "Inferior" refere-se à parte, que está na base da árvore, onde são localizados os coeficientes *wavelet* de mais alta resolução. A implementação é realizada com três subbandas, LH, HH e HL, como se mostra na figura.

Movendo-se da parte superior à inferior na Fig. 4.3, cada bloco tem, no próximo nível mais baixo em cada banda, duas vezes a quantidade de colunas e linhas que tem o bloco no nível superior. Conseqüentemente, cada bloco do nível imediatamente inferior, tem quatro vêzes mais coeficientes que o bloco imediatamente superior. Especificamente, um

coeficiente em uma escala grosseira (chamado de "pai"), exceto nos blocos mais baixos, tem quatros coeficientes que correspondem à mesma localização espacial, chamado de "filhos".

Matematicamente, se o pai está na localização (i, j) os quatros filhos imediatos estão localizados em $(2i-1,2j-1),(2i-1,2j),(2i,2j-1)$ e $(2i,2j)$. Essa relação é mostrada na Fig. 4.4.

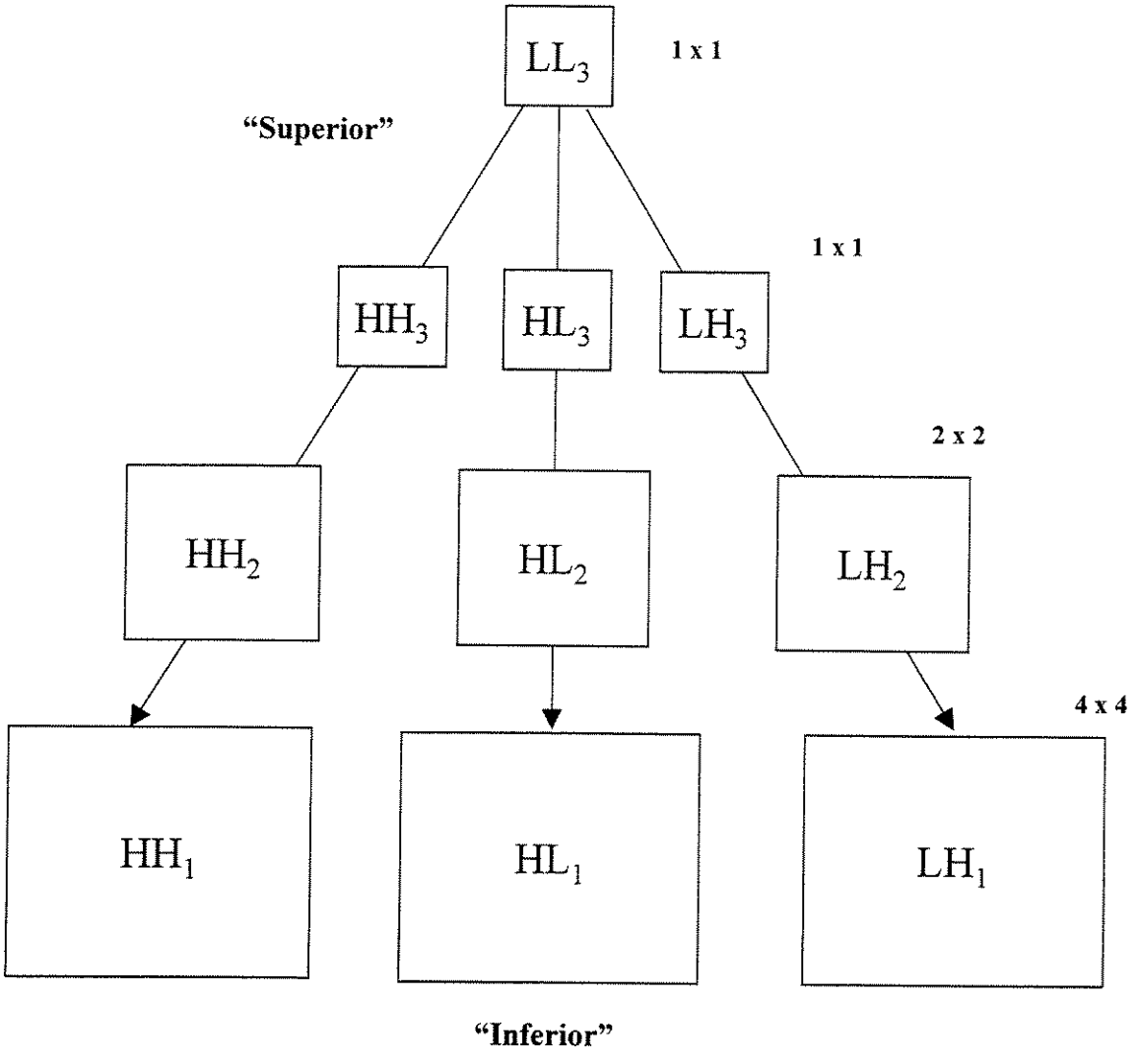


Fig. 4.3: Os blocos da decomposição *wavelet* são organizados nessa estrutura de árvore. A representação corresponde a TW de uma imagem de tamanho 8 x 8.

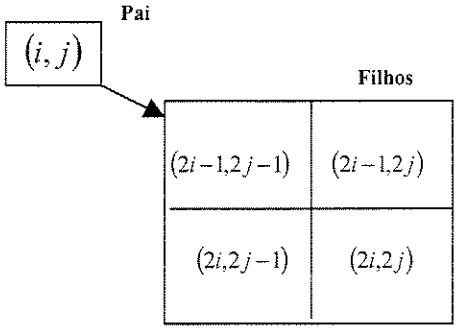


Fig. 4.4: Cada coeficiente do bloco superior tem 4 filhos, no bloco imediatamente inferior.

Conforme frisado anteriormente no Cap.3 é importante se definir para implementação do *zerotree* o que significa dizer que um termo é "significante". Recordando brevemente, um coeficiente *wavelet* ***c*** é dito ser "significante" com relação a um determinado limiar T se $|c| \geq T$; caso contrário tal coeficiente é considerado "insignificante". Um coeficiente insignificante é tomado como sendo nulo. Existem muitos critérios diferentes para a seleção de T. Nessa implementação a partir da representação binária de cada coeficiente *wavelet*, são selecionados, uma seqüência de limiares, sendo todos potências de 2.

No processamento das imagens um conjunto de símbolos é criado, como se mostra na Tabela 4.1. Cada elemento desse conjunto é iniciado com "0" no começo de cada passo comparativo com relação ao limiar T, para indicar que o elemento ainda não foi processado. Neste trabalho é usado um conjunto de cinco símbolos.

Tabela 4.1: Símbolos utilizados para o processamento.

SÍMBOLO	DESCRIÇÃO
POS	Coeficiente *wavelet* que excede o valor de T. É **significante** e **positivo.**
NEG	Coeficiente *wavelet* que excede o valor de T. É **significante** e **negativo.**
IZ	***Zero isolado***. Coeficiente *wavelet* que não excede o valor de T. É **insignificante, mas** tem um **filho significante**.
ZR	**Raiz do *zerotree***. Coeficiente *wavelet* que não excede o valor de T. É **insignificante** e o conjunto de filhos descendentes desse coeficiente (subárvore), são coeficientes **insignificantes**.
ZT	Filho insignificante de um **ZR.**

Somente quatro símbolos, **POS**, **NEG**, **IZ** e **ZR**, são escritos no arquivo de saída (*output file*). A decodificação dos símbolos **ZR** é expandida para preencher com *zeros* a subárvore correspondente, de forma que não exista necessidade de escrever no arquivo de saída os símbolos **ZT** correspondentes. Assim, é possível utilizar dois bits para se representar os quatro símbolos. A utilização de arranjos de símbolos bidimensionais difere das aproximações realizadas pelo EZW [40] e SPIHT [41] e [65], que usam uma lista auxiliar para a determinação das posições dos coeficientes significantes. Nessa implementação os símbolos são escritos no arquivo de saída em cada comparação com o limiar, e assim deve-se manter somente um arranjo de símbolos em memória, em lugar de se ter um arranjo de símbolos para cada valor de limiar.

III.1.2. Manipulação dos Coeficientes Insignificantes

Os coeficientes significantes são fáceis de se controlar: se a magnitude do coeficiente excede o limiar o símbolo é classificado como **POS** ou **NEG,** em dependência do sinal. Os coeficientes insignificantes apresentam mais de um desafio. Antes de se classificar um coeficiente insignificante como: **IZ**, **ZR** ou **ZT** é preciso não só conhecer sobre aquele coeficiente, mas também sobre seus descendentes (filhos, netos, taranetos, etc.) e pai. Isso significa, ter informações sobre os coeficientes que estão acima e abaixo do coeficiente atual nas ramificações da árvore.

Para se obter essas informações são realizadas duas passagens sobre os coeficientes: uma no sentido "Inferior - Superior", para identificar os pais dos filhos significativos, e outra "Superior - Inferior" para identificar as raízes dos *zerotree* (ver Fig.4.3). Dentro de uma subbanda (LH, HH ou HL) a passagem "Inferior-Superior" é primeiramente aplicada. Na Fig.4.5 mostra-se o diagrama desse processo. Começando no bloco mais baixo da subbanda, são identificados os coeficientes significantes e classificados os elementos correspondentes aos arranjos dos símbolos POS ou NEG. Nessa parte do algoritmo tem-se as informações sobre os pais dos coeficientes: nenhuns dos pais desses coeficientes significantes podem ser ZR ou ZT e então eles devem ser codificados como POS, NEG ou IZ. Nesse ponto, todos os pais dos coeficientes significantes são marcados como IZ. Posteriormente, subindo-se através da árvore, esses elementos IZ serão achados e os correspondentes coeficientes significantes POS ou NEG podem ser escritos. O código dessa

parte do algoritmo aparece na função *mark_parents* no arquivo *t zerotree*, e pode ser consultado no Anexo C.

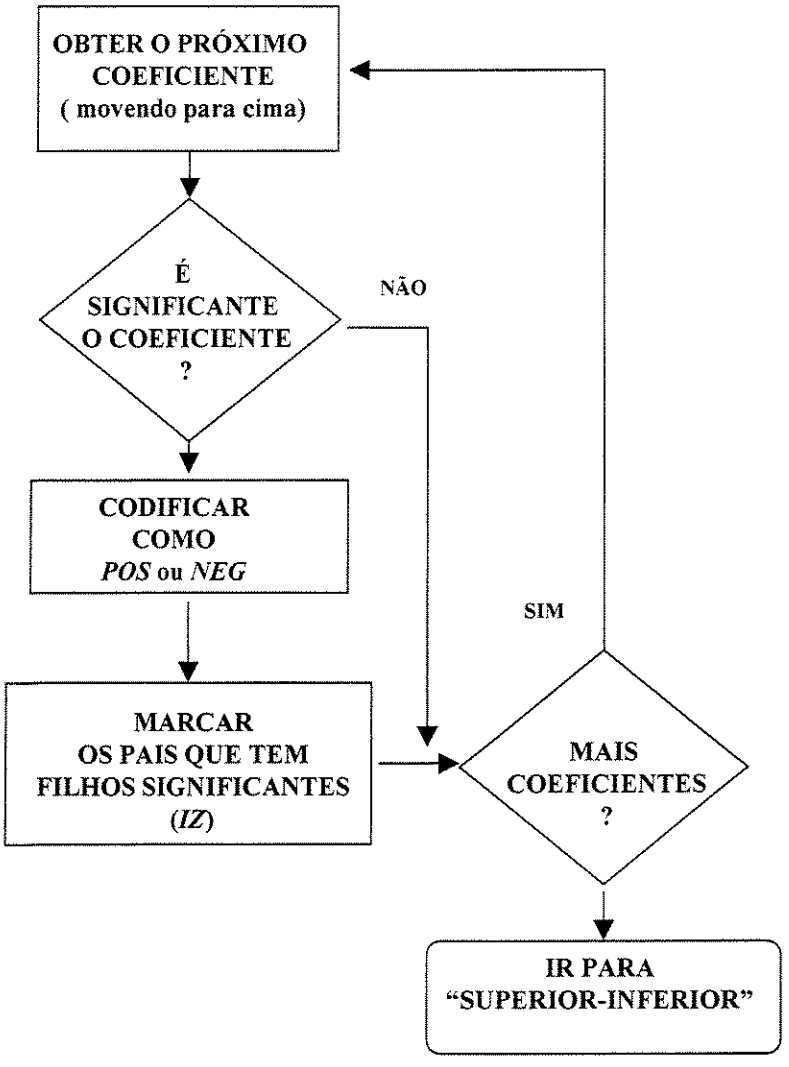


Fig. 4.5: Procedimento "*Inferior-Superior*" utilizado no algoritmo.

Concluindo, na passagem "Inferior-Superior" numa subbanda, todos os "coeficientes significantes" são identificados e codificados como **POS** ou **NEG** e todos os zeros isolados são identificados e codificados como **IZ**. O resto dos coeficientes pode ser: **ZR** ou **ZT**.

A passagem "Superior-Inferior" identifica quais dos coeficientes restantes são **ZR** e quais são **ZT**. Começando no topo da subbanda, é verificado o arranjo de símbolos, para saber qual foi codificado. Como a verificação começa, no topo da subbanda, todo coeficiente que não tenha sido codificado é um ZR. Isso é válido para todos, menos para os coeficientes

que estão no bloco mais baixo da subárvore. Os coeficientes do bloco mais baixo, não têm filhos e por definição não podem ser ZR. Nesse bloco, os coeficientes que não tiverem sido codificados, são classificados como IZ, em lugar de ZR. Quando um ZR é identificado, todos os descendentes filhos (netos, taranetos, etc.) na subárvore são marcados como ZT. O código dessa parte do algoritmo aparece na função *mark_parents* no arquivo *t zerotree* e pode ser consultado no Anexo C.

Na passagem "Superior-Inferior" também são escritos os símbolos no arquivo de saída (*output file*). Quando um símbolo POS, NEG, IZ ou ZR é encontrado, é passado para a saída com o apropriado código de dois *bits*. Os símbolos ZT, não são escritos no arquivo de saída. Na Fig. 4.6 mostra-se o diagrama do processo "Superior-Inferior".

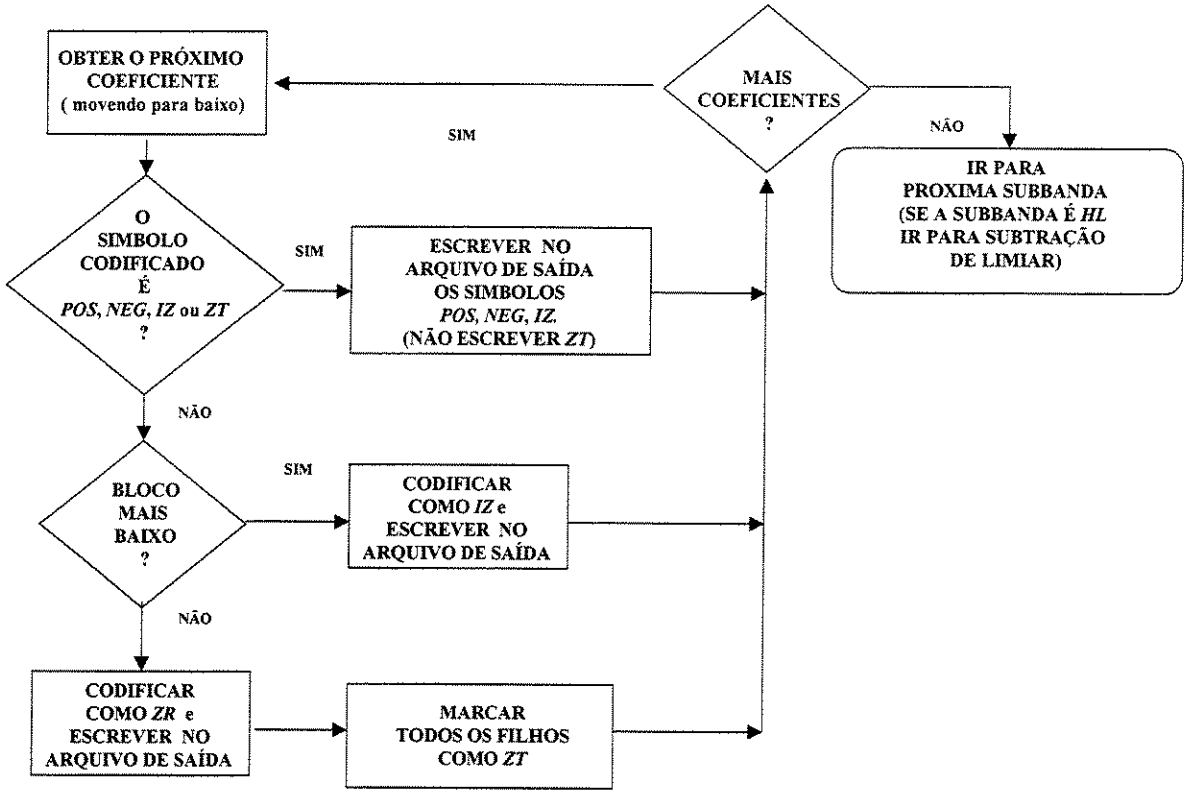


Fig. 4.6: Procedimento "*Superior -Inferior*" utilizado no algoritmo.

IV. Organização em Planos de Bits e Codificação de Entropia.

O último passo dos algoritmos de compressão de imagem, via transformação de sinais é a codificação de entropia. Esse processo é reversível e não há necessidade de aproximações. Após a quantização, os coeficientes assumem valores em um conjunto finito $\{a_i\}$. A idéia é obter um mapeamento M reversível para um novo conjunto $\{b_i\}$ tal que a média de bits por

símbolo seja minimizada. Os parâmetros disponíveis para a busca desse mapeamento M são as probabilidades de ocorrência dos símbolos $a_i, p_i(a_i)$. Se a variável quantizada é estacionária, as probabilidades são fixas e um mapeamento como o de Huffman pode ser utilizado [66].

Se as probabilidades variam durante a codificação, um mapeamento com codificação aritmética pode ser realizado. É importante ressaltar que pelo fato da codificação de entropia não apresentar perda de informação, pode-se associar duas ou mais formas dessa codificação após a quantização. Essa é a idéia básica por trás da qual é aplicada a codificação sem perda em um plano de bits na qual a imagem é dividida em planos de bits cada qual sendo codificado separadamente.

A técnicas de codificação de entropia podem tornar-se mais eficientes tendo em vista que planos de bits mais significativos contêm mais áreas uniformes conforme já foi dito anteriormente.

IV.1. Implementação dos Planos de Bits.

Nesse algoritmo a codificação é realizada a partir da construção seqüencial em planos de bits da TW da imagem. Em nenhuma parte desse algoritmo armazena-se informação, sobre os valores reais dos coeficientes para cada nível de limiar. Porém, subtraindo-se sucessivamente as potências de 2 dos coeficientes, o algoritmo cria uma representação binária dos valores desses coeficientes. A cada passagem comparativa, através do procedimento (*loop*) que contem o limiar é criado um plano de bits (ver Fig. 4.8), começando com os bits mais significativos e terminando com os menos significativos, como se mostra na Fig. 4.7.

No Cap.7, mostra-se a seqüência de planos de bits e símbolos associados obtidos a partir da aplicação desse algoritmo.

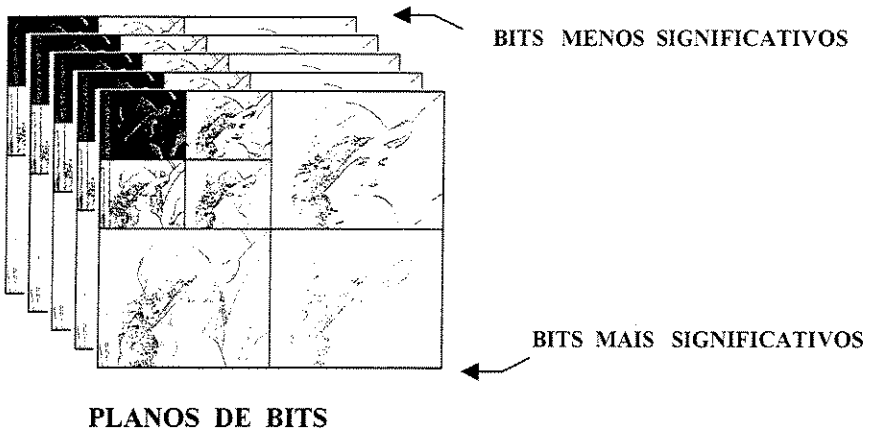


Fig. 4.7: Planos de bits. O Algoritmo produz uma seqüência de planos de bits correspondente a TW da representação binária da imagem.

V. ALGORITMO DE CODIFICAÇÃO

As figuras Fig.4.8 e Fig.4.9 mostram, respectivamente os diagramas de bloco e de fluxos desse algoritmo de codificação.

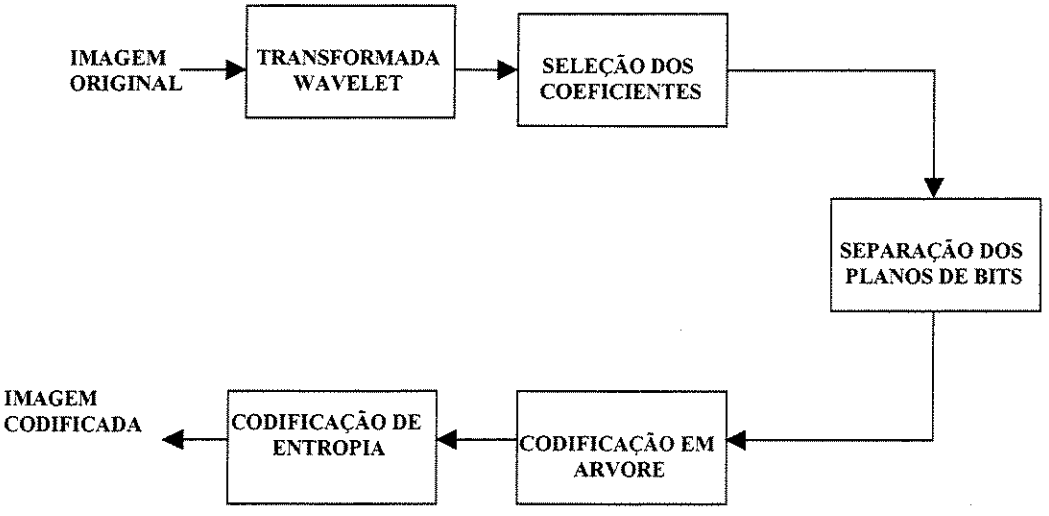


Fig. 4.8: Diagrama de blocos do codificador.

O primeiro estágio do codificador consiste na aplicação da TW. Uma das famílias de *wavelets* estudadas e apresentadas no Cap. 2 pode ser utilizada: *Haar*, *Daubechies*, ou *Spline*, sendo que outros tipos podem também ser colocados nesse algoritmo. O próximo estágio seleciona um valor do limiar T igual à maior potência de 2, que é menor do que a magnitude do maior coeficiente *wavelet*. O *software* desenvolvido para se realizar esse algoritmo mantém, de forma automática um acompanhamento, desses valores máximos dos coeficientes *wavelets* calculados. É freqüente o caso em que o valor do coeficiente passa-

baixas *LL* (o único coeficiente no canto esquerdo superior do arranjo transformado) tem a maior magnitude dentro do arranjo transformado. Esse coeficiente não está localizado em nenhuma das três subbandas *LH*, *HH* e *HL* sendo desconsiderado na determinação do valor máximo transformado.

No arquivo de saída são escritos: o valor do coeficiente passa-baixas, tipo de *wavelet* utilizado, dimensões das linhas (o *software* só trabalha com imagens quadradas) e o valor $log_2(T)$.

O terceiro estágio do algoritmo é realizado dentro do procedimento (*loop*) comparativo com o limiar *T*. O arranjo de símbolos é iniciado com "*0*" no começo de cada passagem pelo *loop*. Para cada subbanda são realizados os procedimentos: "*Superior-Inferior*" e "*Inferior- Superior*" que foram descritos na Secção III.1. As subbandas são processadas na ordem: *LH, HH, HL*. A cada volta (*recall*), os símbolos são escritos no arquivo de saída como parte do procedimento "*Superior-Inferior*". No final do *loop,* o limiar *T* é subtraído de todos os coeficientes significantes, isto é, de todos os coeficientes que tem símbolos POS ou NEG. O limiar é em seguida reduzido pela metade, de modo que o novo valor de *T* é $T/2$. O próximo passo dentro do *loop* é comparar os novos coeficientes *wavelets* com esse novo valor do limiar. Os símbolos resultantes são escritos no arquivo de saída. O processo comparativo dentro do *loop* continua até alcançar a condição $T = 1$.

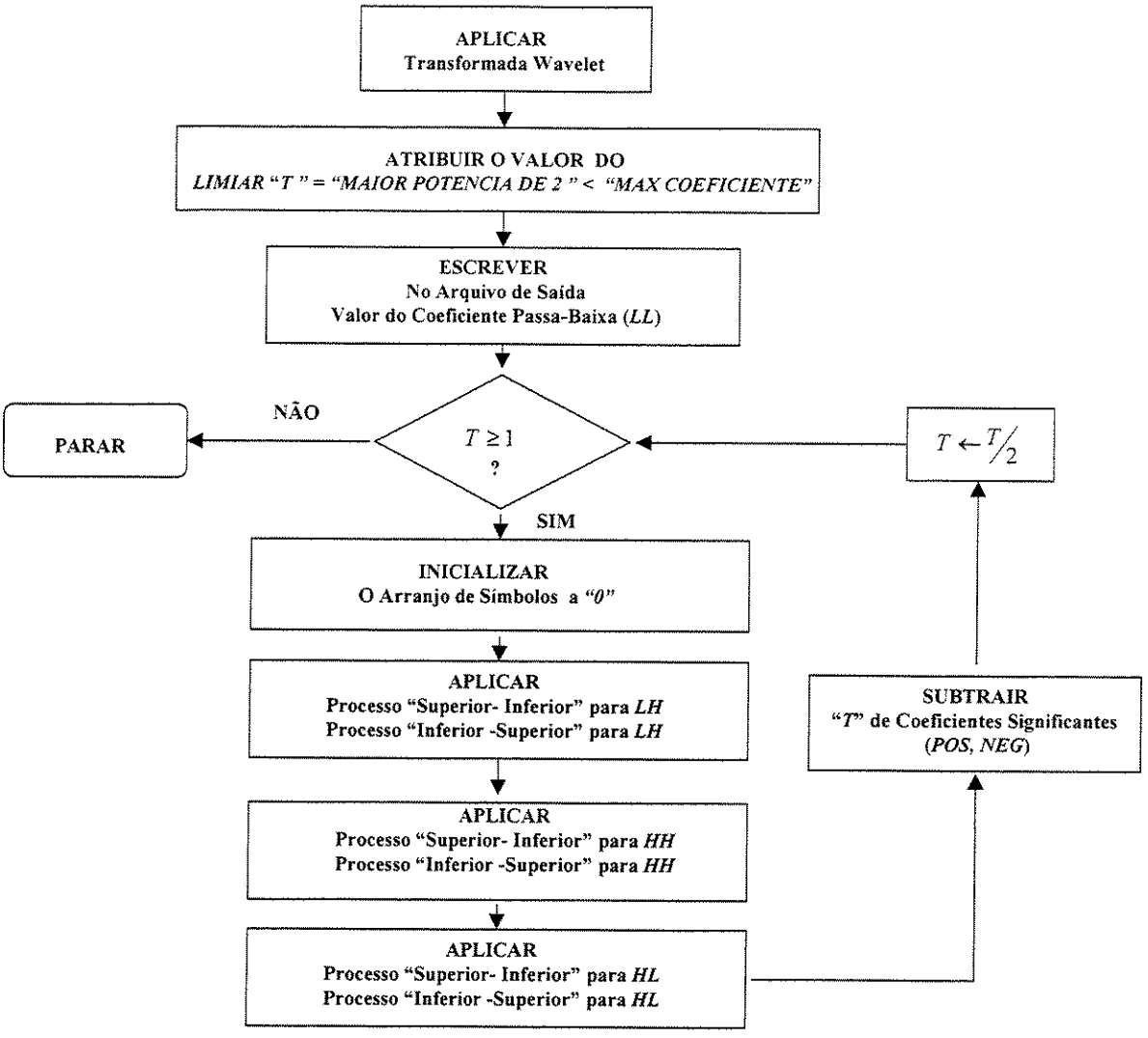


Fig. 4.9: Diagrama de fluxo do codificador.

VI. ALGORITMO DE DECODIFICAÇÃO

O diagrama em blocos desse algoritmo é mostrado na Fig.4.10. O cabeçalho informativo contém:

- Tipo de *wavelet;*
- Número de linhas (imagem quadrada);
- Valor do coeficiente passa baixas;
- $N = log_2(limiar)$;
- Número de planos de bits.

Assim como é feito na codificação usa-se na decodificação um arranjo de símbolos de mesmo tamanho da imagem para se armazenar os símbolos. O arranjo de símbolos é

inicializado no mesmo arquivo *t zerotree* utilizado para a codificação. O arranjo de símbolos é iniciado com *"0"* no começo de cada passagem pelo limiar *T*. Para cada valor do limiar *T* são lidos os símbolos das subbandas *LH*, *HH*, e *HL* desde a parte *"Superior"* até a *"Inferior"* dentro do arranjo de símbolos. Sempre que um símbolo **ZR** é encontrado, esse é expandido, completando com *zeros* toda a sub–árvore, através da atribuição do símbolo **ZT**, para todos seus filhos, no arranjo de símbolos. Essa expansão é realizada utilizando-se o mesmo procedimento *mark_children* do arquivo *zerotree* apresentado no Anexo C.

Os valores da TW se mantêm em um arranjo independente. Esse arranjo da TW é iniciado com *"0"* no começo de cada processo de decodificação. Uma vez que o arranjo de símbolos está completamente cheio para um valor de limiar particular, os valores do arranjo da TW são atualizados somando-se o valor de *T* a cada localização do arranjo correspondente com os símbolos **POS** e subtraindo-se o valor de *T* de cada localização correspondente aos símbolos **NEG**. Portanto, os valores da TW são construídos a partir de sua representação binária, utilizando-se primeiramente os bits mais significativos e progressivamente os menos significativos. A imagem decodificada é obtida aplicando-se a TW inversa após o último plano de bits ter sido construído.

No Cap.7 mostra-se a seqüência de planos de bits obtida no processo de decodificação da imagem Lena, desde o plano mais significativo até o menos significativo, mostrando a decodificação *zerotree* junto com a técnica de transmissão progressiva. Pode-se observar quanta informação é transmitida, aplicando-se a transformada inversa depois que cada plano de bits é construído.

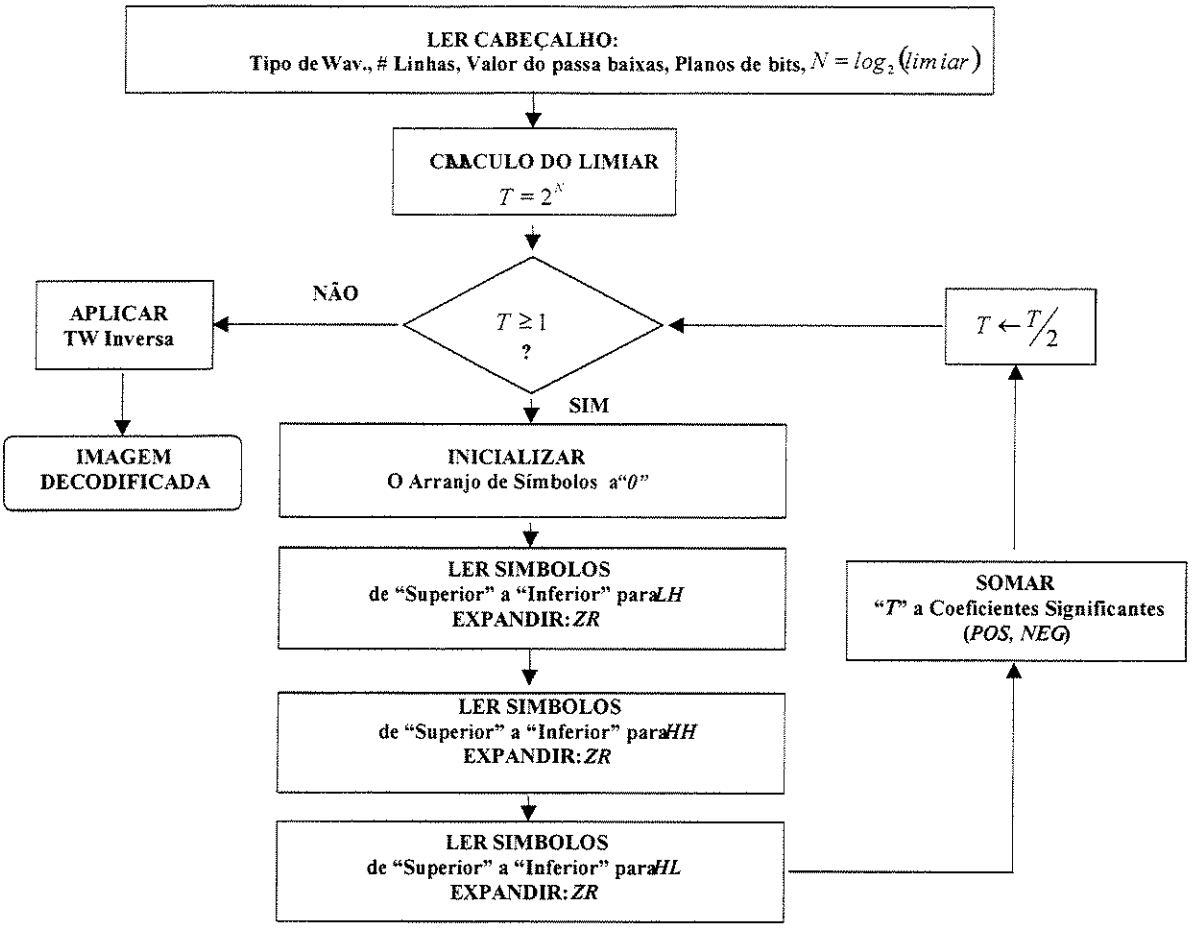


Fig. 4.10- Diagrama de fluxo do decodificador.

CAPÍTULO 5.

Codificação de Vídeo usando Técnicas baseadas na Transformada *Wavelet* com Pré-processamento do Sinal

I. INTRODUÇÃO

Apresenta-se neste capítulo um esquema de um codificador de vídeo baseado na transformada *wavelet* (TW), com compensação local de movimentos. A proposta revela que o processo de codificação do sinal de vídeo se torna mais eficiente, quando um pré - processamento do sinal é realizado, a partir de técnicas que exploram as redundâncias das estruturas de subbandas *wavelets*. O algoritmo também explora a propriedade de escalonabilidade que apresenta a decomposição *wavelet*, onde uma vez feita a decomposição de uma imagem, obtem-se uma outra imagem com uma resolução espacial inferior e três imagens de enriquecimento. Essa característica resultou muito atrativa, na concepção do codificador híbrido proposto, onde a subbanda de baixas freqüências é codificada utilizando-se uma versão modificada do padrão MPEG-2 [67], com compensação local de movimento, que futuramente estará baseada também na transformada *wavelet*. O padrão MPEG-2 foi escolhido, devido ao fato de que apresenta um conjunto de algoritmos e ferramentas que podem ser usados para variadas aplicações em diferentes condições operacionais, isto é, com diferentes taxas, diferentes meios de transmissão ou armazenamento, etc.

O esquema proposto é resultado da adaptação de algumas idéias abordadas por [68, 69, 70, 71, 72, 73] e das pesquisas que sobre esse tema foram realizadas e apresentadas em [74, 75, 76].

Desse modo, no esquema proposto destacam-se três técnicas:

1. Pré-codificação.
2. Codificação Hierárquica ou Escalonabilidade do sinal.
3. Compensação e Estimação Local de Movimentos

Nessa parte da pesquisa, os esforços foram dirigidos fundamentalmente para implementação e teste das técnicas de Pré-codificação e da Codificação Hierárquica a partir

da extensão das técnicas usadas nos esquemas de codificação de imagens [75, 76], para a codificação do sinal de vídeo com algoritmos baseados na TW.

II. Descrição dos Sistemas Implementados para Simulação

O sistema implementado neste trabalho é mostrado no diagrama da Fig.5.1. A seguir é feita uma descrição detalhada de cada um dos blocos integrantes do sistema.

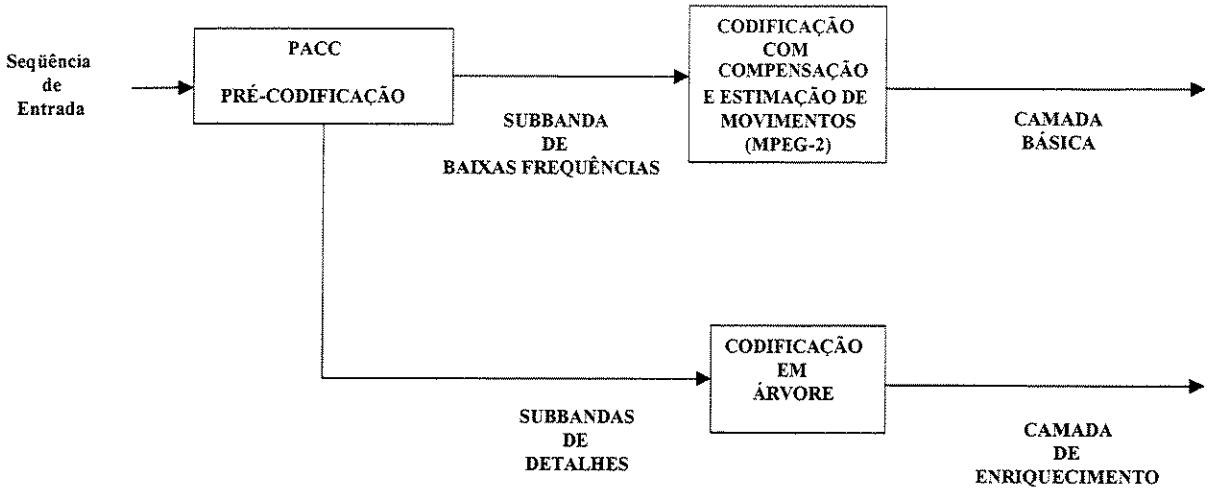


Fig. 5.1: Diagrama em blocos do codificador.

II.1. Pré -Codificação PACC

A introdução e aplicação dessa técnica baseiam-se, nos conceitos de codificação condicional, particionada e agregada (PACC), proposta por D. Marpe e H. L. Cyon (1997) [76].

A idéia, foi desenvolvida a partir da observação da propriedade de localização espaço-freqüência da TW de imagens, o qual levou a criar um mecanismo que explora as redundâncias características da representação *wavelets* [75, 76].

Aplicando-se PACC, neste projeto, primeiramente os dados são divididos, em diferentes subfontes ("partição"). No segundo passo os dados ou conjuntos particionados são correlacionados e aproximados ao longo de subbandas descendentes ("agregação"). Mediante o uso de modelos baseados na probabilidade condicional, pode-se recuperar, as correlações entre as estruturas construídas, assim como das subfontes de dados, o qual pode ser usado pelo codificador final ("codificação condicional").

A maioria dos esquemas codificadores por transformadas publicados na literatura, aplica combinadamente códigos de comprimento variável, codificação de *run-length* e preditiva,

para se remover a correlação dos coeficientes quantizados. Esses métodos são conhecidos por oferecer uma boa relação de compromisso entre eficiência e complexidade. Na procura por obter-se uma melhora na eficiência da codificação, modelos mais complexos e sofisticados, estão sendo desenvolvidos.

Assumindo-se um alto grau de dependência estatística, no conjunto de dados, uma aproximação para um modelo de ordem N, poderia ser realizada, mas para isso se necessitaria estimar a função de distribuição de probabilidades (PD) de M^{N+1} valores diferentes do vetor aleatório $\mathbf{s} = \{s_0,..., s_N\}$, em um alfabeto de tamanho M, muito maior que N, o que faz que esse modelo não seja prático de modo geral. Torna-se necessário, realizar algumas modificações nesse modelo, para manter um grau máximo de dependência estatística entre os dados.

O primeiro passo nessa modificação é chamado de Partição, onde a fonte é dividida para reduzir o tamanho do alfabeto M. Na. Fig. 5.2 mostra-se o processo, proposto neste trabalho, na pré-codificação PACC. Observe-se no estágio inicial a existência de três subfontes:

- **Mapa de Significância**: indicando a alocação dos coeficientes significantes.
- **Mapa de Magnitude**: para armazenar os valores absolutos dos coeficientes significantes
- **Mapa de Sinais**: que contém a informação do sinal e fase dos coeficientes *wavelets*.

Essa estrutura para o tratamento e organização dos dados, como conjunto particionando, é baseada nas seguintes propriedades estatísticas da transformada *wavelets*,

- Similaridade entre subbandas de diferentes escalas,
- Decaimento da magnitude dos coeficientes *wavelets* entre subbandas à medida que se refina a escala de resolução.

Um estudo mais detalhado sobre essas propriedades estão no Cap. 2 [77].

O **segundo passo**, foi chamado de **agregação,** baseado nas idéias propostas em [40, 42, 78, 79,], onde os coeficientes insignificantes podem ser aproximados ao longo de subbandas descendentes chamadas de árvore de zeros.

No último estágio do pré-codificador proposto, é fornecido um modelo apropriado para o processo de codificação hierárquica.

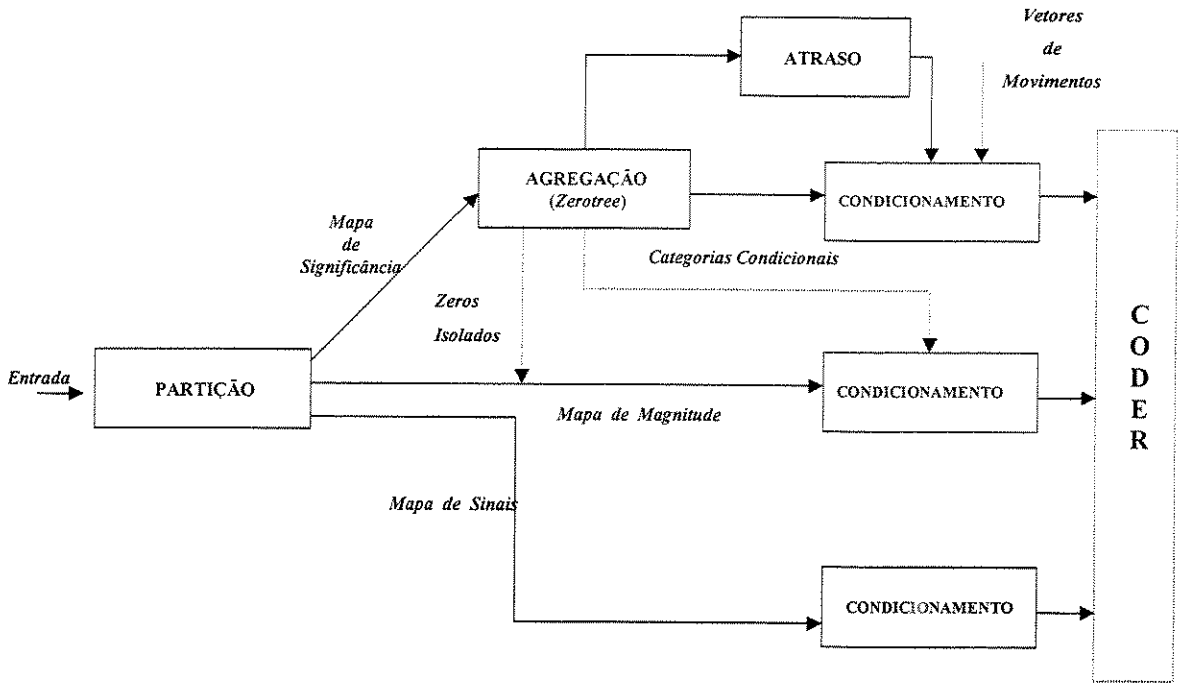


Fig. 5.2 Pré-codificador PACC.

II.1.1. PARTIÇÃO

A base teórica da "partição" é dada por dois teoremas, que foram introduzidos inicialmente em contextos diferentes [80] e descobertos mais tarde para serem usados na codificação de imagens baseada na transformada *wavelet* [81]. O primeiro, a partir de um dos teoremas indica que a taxa da entropia de uma fonte pode ser reduzida dividindo-se a fonte em subfontes desconexas não vazias [80].

No cenário de codificação, esse princípio da partição da fonte tem uma aplicação óbvia: a imagem transformada é dividida, de acordo com uma estrutura de subbandas, nas subfontes $\hat{c}_{l,k}$ com diferentes densidades de probabilidades. Embora esse primeiro passo da partição possa já diminuir a taxa total de entropia, são adicionadas mais duas etapas, para se obter um intervalo adaptativo de particionamento, caracterizado pelo seguinte teorema:

Teorema 4.1: Particionamento de intervalo adaptativo.

Seja o intervalo dinâmico de uma fonte ***S*** dada por $A = \bigcup_i A_i$, onde A_i são subconjuntos, desconexos e não vazios de A e definindo-se as subfontes $\widetilde{S} = \{s \in S | s = \alpha, \alpha \in A_i\}$ e o

conjunto de indicadores $\chi = \{x_k | x_k = i, \ se \ \ s_k \in A_i\}$, então a taxa total da entropia é dada por:

$$R(S) = \sum_i R(\widetilde{S}_i) + R(\chi) \tag{5.1}$$

O intervalo de particionamento adaptativo não aumenta a taxa de entropia (que é a interpretação essencial da Eq.(5.1), mas permite recuperar a informação. Dividindo-se o intervalo segundo a significância dos coeficientes, resultará em uma subfonte $\hat{c}_{l,k}^{sign}$ dos coeficientes significantes, sendo a fonte que contém a parte da informação dessa partição adaptativa chamada de mapa de significância $\chi_{l,k}$. Então, finalmente na Eq.(5.1), representa-se um intervalo da partição adicional, que represente o sinal dos coeficientes significantes,

$$R(\hat{c}_{l,k}) = R(\chi_{l,k}) + R(\hat{c}_{l,k}^{mag}) + R(\varsigma_{l,k}) \tag{5.2}$$

onde o **mapa de amplitude** $\hat{c}_{l,k}^{mag}$ é a subfonte que contem a amplitude dos coeficientes significantes e o mapa do sinal $\varsigma_{l,k}$ armazena a informação relevante do sinal.

Na Eq.(5.2), observa-se que devido à substituição do $\hat{c}_{l,k}$, por dois mapas com indicadores de valores binários $\chi_{l,k}$ e $\varsigma_{l,k}$ e pelo mapa de amplitude, $\hat{c}_{l,k}^{mag}$ não altera o limite mais baixo da taxa atingível pelo codificador, mas o que é mais importante, sugere uma maneira de aproximação em direção a esse limite.

O processo de particionamento permite uma melhor utilização da independência de diferentes subfontes, quer sejam de diferentes tipos quer sejam de diferentes conteúdos de freqüências.

II.1.2. AGREGAÇÃO (*ZEROTREE*)

Para o processo de codificação de vídeo duas exigências são essenciais:

- Após a quantização dos intraquadros (quadros tipo I) somente uma pequena fração dos coeficientes deve ser **não zero** (significantes)
- A saída do esquema da predição temporal deve ter um erro de predição, com uma baixa energia, resultado da representação dos poucos coeficientes *wavelet* **não zero** (significantes) [82].

Embora essas exigências nem sempre sejam estritamente cumpridas, especialmente na presença de cenas com grande movimento, é óbvio, que é necessário um método eficiente de codificação dos coeficientes **insignificantes** (zeros) quantizados. A idéia é definir uma árvore de símbolos nulos ou zeros (*zerotree*). Dessa forma, a raiz pode ser nomeada como o fim do bloco relativo a toda a árvore de zeros.

No Cap.4 foi discutida e implementada a codificação *zerotree.*

II.1.3. CODIFICAÇÃO CONDICIONAL

Para a codificação dos mapas com indicadores binários pode-se impor ao sistema métodos de codificação hierárquica.

II.1.3.1. CODIFICAÇÃO HIERÁRQUICA

A codificação hierárquica é uma classe de técnicas em que a imagem é codificada em vários níveis de qualidade ou resolução. O termo "hierárquica" é utilizado, pois os dados referentes à imagem são organizados em ordem de importância, ou seja, em vários níveis de hierarquia, onde cada nível corresponde a uma imagem reconstruída com um determinado nível de qualidade ou resolução. Nas técnicas que utilizam codificação hierárquica, algumas das características desejáveis são listadas a seguir:

- A reconstrução da imagem em determinado nível deve aproveitar ao máximo as informações já transmitidas para os níveis inferiores.
- O escalonamento dos dados em níveis de hierarquia deve prejudicar o mínimo possível o ganho de compressão, em relação ao obtido quando não há o compromisso de se possibilitar a reprodução da imagem em vários níveis de qualidade ou resolução.
- Devem existir algoritmos de codificação e decodificação relativamente rápidos e de fácil implementação em *hardware*.

Uma das aplicações para essa técnica é transmissão progressiva, e outra aplicação é em ambientes de múltiplo uso, em que as imagens podem ser apresentadas em telas de diferentes resoluções. Nesse caso, a quantidade de dados para se alcançar o próximo nível, ao contrário da transmissão progressiva, cresce exponencialmente.

II.1.3.2. HIERARQUIAS DE RESOLUÇÃO VARIÁVEL

Nas técnicas de hierarquia de resolução espacial variável, é comum a interpretação do conjunto de níveis como uma estrutura piramidal, em que a base representa a imagem com resolução total. Por exemplo, cada nível da pirâmide pode ser formado tomando-se a média de blocos de 2x2 *pixels*, como apresentado no exemplo da Fig.5.3. Nessa figura são mostradas as conexões entre os *pixels* de mesma região espacial. Geralmente, o *pixel* do nível imediatamente superior é utilizado como estimador na codificação dos quatro *pixels* de mesma posição espacial que, na Fig. 5.3, estão a ele conectados.

Os valores dos pontos de cada nível da pirâmide podem ser obtidos de várias outras formas. Por exemplo, pode-se dizimar a imagem nas duas direções, horizontal e vertical. Dessa forma, só os outros três *pixels* precisam ser enviados. Essa técnica apresenta a vantagem de manter a mesma quantidade de pontos da imagem original, entretanto a dizimação introduz *aliasing* que prejudica a visualização dos níveis superiores. Além disso, o ponto escolhido pode não representar bem a região da qual foi tomada.

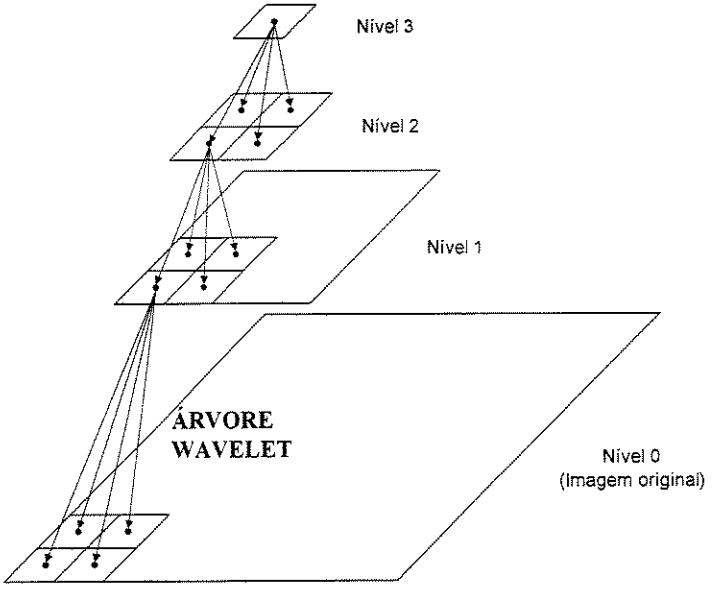


Fig. 5.3: Exemplo de codificação hierárquica com estrutura piramidal.

Para se diminuir o efeito de *aliasing* pode-se tomar a média dos quatro pontos como representante da região, e a informação complementar é obtida, por exemplo, tomando-se a diferença entre os *pixels* e a média desses. Outras técnicas semelhantes podem ser também utilizadas, como a pirâmide de somas reduzidas, a pirâmide de diferenças reduzidas, entre outras [83].

A transformada *wavelet*, de forma natural possibilita a codificação hierárquica. Isso porque a subbanda de freqüências mais baixas representaria o nível imediatamente superior na pirâmide. A essa subbanda é novamente aplicada a divisão em subbandas até que se obtenha o número de níveis desejados.

Quando se trata de vídeo pode-se realizar, além do escalonamento espacial, um escalonamento temporal, em que a taxa de quadros por segundo varia para cada nível da hierarquia.

III. FUNDAMENTAÇÃO DO ALGORITMO

Na Fig.5.5 mostra-se, o diagrama do codificador. O sistema foi implementado, utilizando-se a linguagem C como ferramenta, e uma versão modificada do codec baseado no Padrão MPEG-2, a partir dos códigos de programas executáveis de domínio publico. (www.mpeg.org/mpeg).

O primeiro bloco nomeado PACC compreende as operações indicadas na Secção II, e os códigos dos programas desenvolvidos podem se consultados no Anexo C. A implementação foi realizada no domínio, *wavelet* a partir da aplicação da TW bidimensional com extensão separável (ver Cap. 4), com 3 níveis de decomposição gerando-se um total de 10 subbandas. Os filtros utilizados são biortogonais de comprimento impar e seus coeficientes são mostrados na **Tabela 5.1.** A subbanda de baixas freqüências é codificada de modo independente da subbanda de detalhes.

Tabela 5.1: Coeficientes filtros *spline* biortogonais com comprimentos p=9 e q=7

n	0	±1	±2	±3	±4	±5	±6	±7	±8
$2^{-1/2}h[n]$	0.37828	-0.023849	-0.110624	0.377402	0.852699	0.377402	-0.110624	-0.023849	0.037828
$2^{1/2}\tilde{h}[n]$	0.064359	-0.040689	-0.418092	0.788486	-0.418092	-0.040689	0.064359	0	0

No codificador implementado, a seqüência digital é codificada hierarquicamente em duas camadas utilizando-se a técnica de escalonabilidade SNR. A primeira camada básica é essencial para decodificação, enquanto a camada de enriquecimento representa informação adicional do detalhamento para se melhorar a qualidade da seqüência reconstruída.

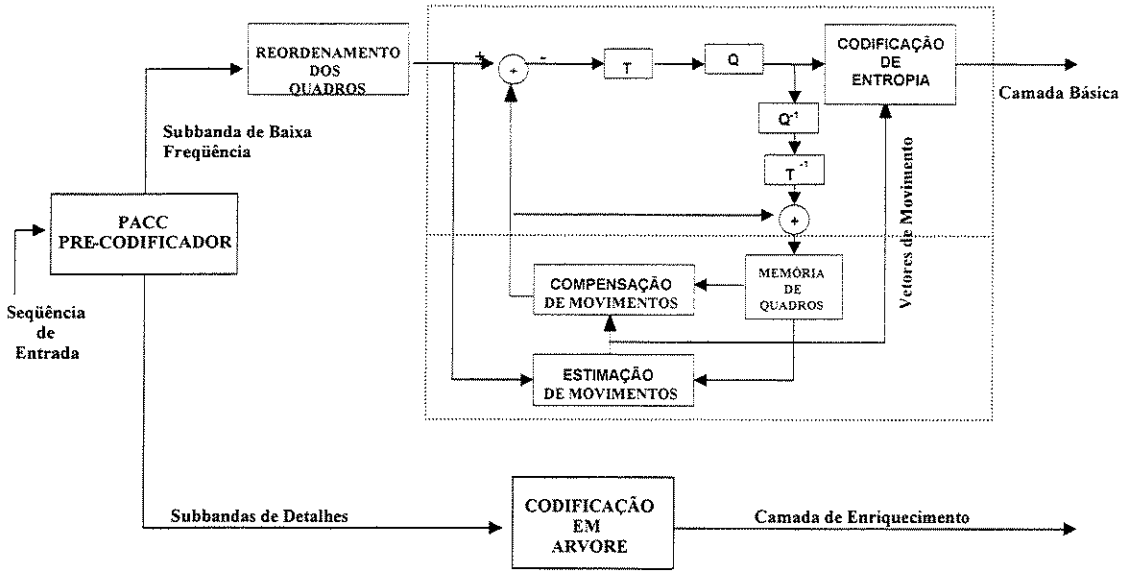


Fig. 4.11: Codificador.

III.2. CODIFICADOR COM ESCALONABILIDADE SNR

Nessa técnica, a primeira camada, que contém coeficientes grosseiramente quantizados, é essencial para a decodificação e será chamada camada de alta prioridade. Por outro lado, a segunda camada contém dados para refinamento desses coeficientes e corresponde a camada de baixa prioridade. Camada de baixa prioridade tem como finalidade melhorar a relação sinal ruído (SNR – *Signal to Noise Ratio*) da seqüência reconstruída, esse esquema de separação em camadas é chamado, também, requantização.

De modo geral, no processo de predição de quadros, que envolve estimação e compensação de movimento, ocorre que cada quadro da seqüência de entrada pode ser codificado de dois modos: codificação intraquadro ou codificação interquadros. Nesse último só é codificada a diferença entre o quadro e a sua predição. A predição pode ser feita utilizando-se o quadro de referência anterior ou dois quadros de referência, um anterior e outro posterior. No primeiro caso o quadro codificado é chamado de predito (quadro tipo P), enquanto que no segundo é chamado de quadro interpolado (quadro tipo B). Somente os quadros intracodificados e os preditos podem ser utilizados como referência na etapa de predição.

O primeiro quadro de uma seqüência é sempre intracodificado, visto que ainda não há quadros de referência disponíveis no preditor. Esse quadro é dividido em blocos de 8 x 8

pixels, e a cada bloco é aplicada a transformação. Em seguida os coeficientes são quantizados. O próximo passo é o processo de triagem dos coeficientes e a forma de escolha dos coeficientes que são enviados em cada canal (de alta ou baixa prioridade). Os coeficientes passam então por um processo de codificação de entropia que envolve codificação de Huffman e codificação *runlength*. Finalmente, os coeficientes são enviados através de seus respectivos canais.

Na etapa de triagem também é feita a escolha dos dados que serão utilizados na etapa de predição. Esses dados podem ser referentes às camadas de alta e baixa prioridade ou somente referentes à camada de alta prioridade, como foi no caso do presente projeto.

Os coeficientes que serão utilizados na etapa de predição passam, então, por um processo de quantização inversa seguido de uma transformação inversa. Nesse ponto, tem-se um quadro reconstruído que é usado como referência, por outros quadros, nas etapas de estimação e compensação de movimento.

O processo que foi descrito é para codificação *intraframe* de um quadro. Caso a codificação seja *interframe* o processo é semelhante. Nesse caso, apenas a diferença entre o quadro atual e a sua predição é codificada. Se o quadro codificado é do tipo B não é necessário que ele seja recuperado no codificador, já que ele não é utilizado como referência na etapa de estimação e compensação de movimento.

Os vetores de deslocamento resultantes da etapa de estimação e compensação de movimento são codificados utilizando-se código de Huffman (codificação de entropia) e em seguida são enviados no canal de alta prioridade.

Nesse caso, os dados a serem codificados (o quadro atual ou a diferença entre este e sua predição) passam pela etapa de transformação e os coeficientes resultantes são quantizados grosseiramente por Q_1. Esses coeficientes quantizados passam pela etapa de codificação de entropia e são enviados no canal de alta prioridade. Esses mesmos coeficientes são recuperados e a diferença entre esses e os coeficientes originais é calculada. Essa diferença é quantizada em Q_2 e os dados resultantes são enviados, após a codificação de entropia, através do canal de baixa prioridade. Se a etapa de predição é baseada nos dados de alta e baixa prioridade, esses dados são recuperados na quantização inversa de Q_2, somados com a saída da quantização inversa de Q_1 e essa soma é encaminhada para a etapa de

transformação inversa. Desse ponto, os coeficientes reconstruídos alimentam o preditor. A escolha dos dados utilizados na estimação do preditor.

CAPÍTULO 6.

Codificador Proposto com Estimação de Movimento no Domínio *Wavelet*.

I. INTRODUÇÃO

Neste capítulo descreve-se finalmente o codificador de vídeo proposto. Mostram-se os fundamentos que levaram ao esquema final onde se propõe uma técnica para a estimação de movimento no domínio *wavelet* a partir da propriedade da variância com o deslocamento, junto com o modelo de pré-processamento conhecido como PACC (*Partition, Agregation and Condition Coding*).

Nas seções anteriores foi discutida a capacidade de representação multiresolucional, da transformada *wavelet*, a partir da qual pode-se aplicar as técnicas de escalonabilidade e de transmissão progressiva, que resultam ser interessantes, na codificação e no processamento do sinal de vídeo.

A compensação de movimento é uma técnica eficiente usada em esquemas de processamento de sinais para se realizar a predição temporal. Existem vários estudos, nos quais é proposta uma predição dos coeficientes que seja realizada diretamente no domínio *wavelet*, aplicando-se compensação de movimento banda a banda [84, 85, 86, 87, 89, 90, 91]. Entretanto, existe uma grande diferença entre os coeficientes *wavelets* da imagem original e da imagem deslocada de 1 pixel. Essa propriedade de ser variante com o deslocamento (*shift-variant*) acontece freqüentemente em torno das bordas da imagem, sendo que desse modo a compensação de movimento no domínio *wavelet* pode ser ineficiente [92]. Por conseqüinte, novos métodos devem ser estudados para se dominar essa propriedade de variante com o deslocamento a fim de se poder aplicar as atrativas características da transformada *wavelet* no processamento de vídeo.

A seguir descreve-se a organização das secções. Na Seção II, apresenta-se a técnica de estimação e compensação de movimento, no domínio espacial e no domínio *wavelet*. A seguir na Seção III, se propõe um esquema de estimação e composição de movimentos que

permite, dominar a propriedade de variância com o deslocamento (*"shift variant"*) da transformada *wavelet* discreta (TWD). Em seguida, são apresentados estudos comparativos de algumas técnicas de estimação e compensação de movimentos para a codificação de sinais de vídeo baseada na TW. Na Seção IV apresenta-se o codificador proposto.

II. Estimação e Compensação de Movimentos Convencional para a Codificação baseada na TW

II.1. Estimação de Movimentos no Domínio Espacial

Em muitos esquemas de codificadores de vídeo a estimação e codificação são realizadas no domínio espacial, assumindo-se no modelo, que os movimentos são translacionais [9-Cap.6]. A técnica baseada em blocos (*block -based*), a qual é utilizada em vários padrões de compressão de vídeo, como por exemplo, MPEG-2, H261, H263, entre outros [84], pressupõe que a imagem é composta de objetos rígidos com movimentos translacionais. Contudo, as técnicas de estimação e compensação de movimentos que usam esquemas baseados em blocos, muitas vêzes produzem, descontinuidades entre os movimentos dos blocos compensados, desde que os vetores de movimentos vizinhos não sejam correlatos. Quando a TW é aplicada sobre sinais residuais, da compensação de movimentos, a descontinuidade entre blocos acentua-se nos coeficientes da TW dos sinais de alta freqüência e dessa forma, a eficiência do processo de codificação diminui. Por conseguinte, torna-se necessária uma efetiva redução da descontinuidade dos blocos nos sinais residuais quando se deseja obter uma codificação altamente eficiente [48].

Existem outras propostas, onde a estimação e compensação do movimento são realizadas nos domínios espacial e *wavelet,* respectivamente. Uma vez que, a compensação de movimentos é realizada no domínio *wavelet* a TW não gera sinais de altas freqüências nos blocos fronteiros (*block boundary*). Porém, os vetores de movimentos obtidos no domínio espacial, podem não ser ótimos para se compensar os movimentos no domínio *wavelet* [93, 94].

II.2. Estimação de Movimentos no Domínio *Wavelet*

O processo de dizimação da TW converte o processo em variante com o deslocamento (*shift variant*). Uma vez que a TW é variante com o deslocamento, essa técnica não é eficiente na obtenção dos vetores de movimento em algoritmos que utilizem técnicas de

casamento de blocos, no domínio *wavelet*. O sinal da banda baixa é usualmente suave e as diferenças na banda baixa, entre coeficientes do sinal original e do sinal deslocado são pequenas. Assim é possível se estimar os coeficientes da banda baixa dos sinais deslocado e original com um pequeno erro de predição. Contudo, existe uma grande diferença entre os coeficientes da banda alta dos sinais deslocado e do original. Tal fenômeno acontece com mais freqüência nas bordas da imagem. O sinal diferença das bandas altas depende da quantidade de deslocamentos e de filtros de decomposição utilizados. A predição dos erros de sinais das bandas altas, no processo de estimativa dos vetores de movimento, torna-se muito dificultosa no domínio *wavelet*, quando a técnica de casamento de blocos na forma convencional é aplicada.

Diversos métodos para se realizar a estimação e compensação de movimentos no domínio *wavelet* tem sido propostos [5, 6, 8, 9, 13-Cap.3]. Dentre esses métodos, o de estimação de movimento, dos coeficientes *wavelets* banda a banda (*band to band*) não é eficiente, uma vez que o esquema com a TW tem a propriedade de ser variante com o deslocamento. Existe outra forma de se realizar a estimação de movimento apenas no sinal das bandas baixas, onde a compensação de movimentos do sinal das bandas altas é realizada com os correspondentes vetores de movimentos encontrados nos sinais de bandas baixas.[14-Cap.3].

III. Método de Deslocamento da Banda Baixa para Estimação e Compensação de Movimentos no domínio Wavelet

III.1. Método de Deslocamento da Banda Baixa [95].

Na Fig.6.1 apresenta-se um esquema, que permite explorar a propriedade de variância com o deslocamento, baseado no método conhecido como deslocamento da banda baixa (*low-band-shift -LBS*). No esquema da Fig.6.1, o método é representado com 3 níveis de decomposição da transformada *wavelet* unidimensional. O *i*–ésimo nível do sinal de banda alta da entrada *f(x)* é representado por $H_f^{(i)}(d,x)$, onde *d* indica o número de deslocamentos no domínio espacial e *x* é a posição da subbanda. Por exemplo, o sinal original e os sinais deslocados são decompostos em seus sinais de bandas altas e baixas. O primeiro nível do sinal de banda alta gerado pelo sinal original é designado como $H_f^{(0)}(0,x)$, e o sinal de banda alta proveniente do sinal deslocado de 1 (um) pixel é denotado como $H_f^{(1)}(1,x)$.

No processo de decomposição hierárquica, do esquema proposto, o sinal de banda baixa é decomposto sucessivamente, tal como foi realizado no primeiro nível. Os detalhes desse processo foram apresentados no Cap. 2. Se a decomposição é realizada até o terceiro nível, um total de 8 sinais de banda baixa e 14 sinais de banda alta será gerado como é mostrado na Fig. Fig.6.1. Por exemplo, seja $f_s(n)$ o sinal deslocado de $f(n)$ em *d*- pixel,

$$f_s(n) = f(n+d) \tag{6.1}$$

Em cada sub-banda do sinal deslocado, $f_s(n)$, pode ser representado da seguinte forma:

$$\begin{aligned} H_{f_s}^{(1)}(n_1) &= H_f^{(1)}\left(d\%2, n_1 + \left\lfloor \frac{d}{2} \right\rfloor \right), \\ H_{f_s}^{(2)}(n_2) &= H_f^{(2)}\left(d\%2^2, n_2 + \left\lfloor \frac{d}{2^2} \right\rfloor \right), \\ H_{f_s}^{(3)}(n_3) &= H_f^{(3)}\left(d\%2^3, n_3 + \left\lfloor \frac{d}{2^3} \right\rfloor \right), \end{aligned} \tag{6.2}$$

onde $H_{f_s}^{(1)}(n_1)$ é o sinal de banda alta do primeiro nível de decomposição de $f_s(n)$,

$H_{f_s}^{(2)}(n_2)$ é o sinal de banda alta do segundo nível de decomposição de $f_s(n)$,

$H_{f_s}^{(3)}(n_3)$ é o sinal de banda alta do terceiro nível de decomposição de $f_s(n)$,

$L_{f_s}^{(3)}(n_3)$ é o sinal de banda baixa do terceiro nível de decomposição de $f_s(n)$,

x % y é a operação módulo, na qual x *é* o *módulo* de y

$\lfloor x \rfloor$ é o operador utilizado para designar o valor inteiro menor ou igual que x

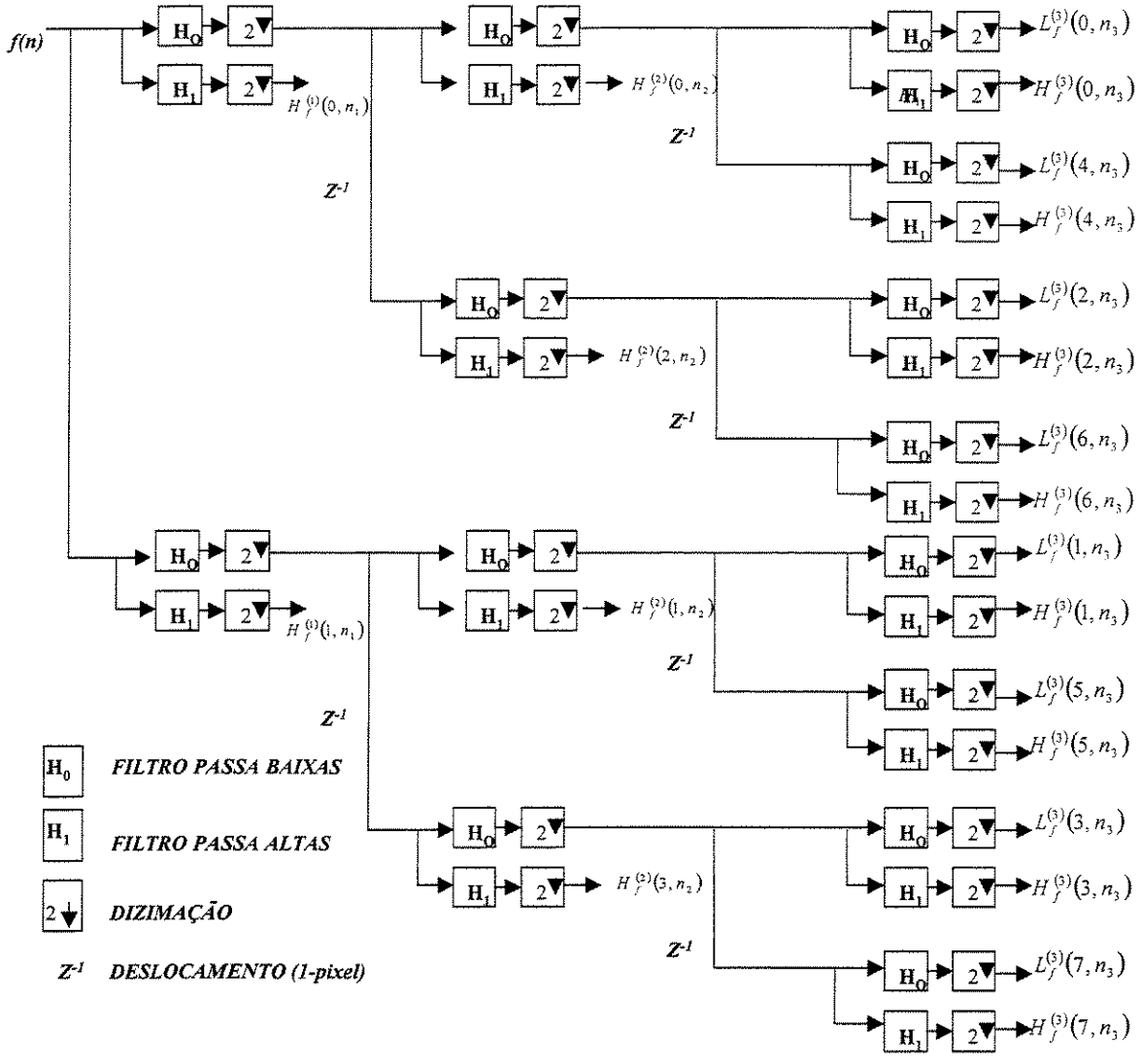


Fig.6.1: O Método LBS em três níveis de decomposição *Wavelet* unidimensional.

III.2. ESTIMAÇÃO E COMPENSAÇÃO DE MOVIMENTOS DE "BLOCOS- *WAVELET*" UTILIZANDO O MÉTODO DE DESLOCAMENTO DA BANDA BAIXA (LBS).

Para se reduzir a redundância interquadro na codificação de vídeo, geralmente assume-se que os objetos têm movimentos translacionais. Embora esse modelo seja restrito, ele se justifica pelo fato de que movimentos complexos (e.g.,deformação) podem ser decompostos em uma soma de movimentos translacionais. Assim, algoritmos de casamentos de blocos são usualmente aplicados para se estimar os vetores de deslocamentos dos blocos de imagens no domínio espacial [84, 96]. Os coeficientes da TW de uma imagem têm sua própria informação de localização espacial, orientação e banda de freqüência. Por conseqüinte, os coeficientes da TW podem ser agrupados em blocos de dados de acordo com a localização espacial, cada um deles correspondendo a um bloco de imagem no domínio espacial. Os coeficientes *wavelets* de cada bloco raiz, na banda baixa

são reagrupados, para se formar os "blocos *wavelets*", como mostra a Fig. 6.2. A finalidade dos "blocos *wavelets*" é proporcionar uma associação direta, entre os coeficientes *wavelets* e a sua representação espacial na imagem. Em cada bloco são incluídos coeficientes relativos a todas as escalas e orientações. Por exemplo, se três níveis de TW são aplicados a uma imagem, como é mostrado na Fig. 6.2, um bloco *wavelet* de 16 x 16 estará constituído de:

- três blocos de 8 x 8, provenientes do primeiro nível de decomposição correspondentes às sub-bandas $HL^{(1)}$, $LH^{(1)}$, $HH^{(1)}$,
- três blocos de 4 x 4, provenientes do segundo nível de decomposição correspondentes às sub-bandas $HL^{(2)}$, $LH^{(2)}$, $HH^{(2)}$,
- quatro blocos de 2 x 2, provenientes do terceiro nível de decomposição correspondentes às sub-bandas $LL^{(2)}$ $HL^{(2)}$, $LH^{(2)}$, $HH^{(2)}$.

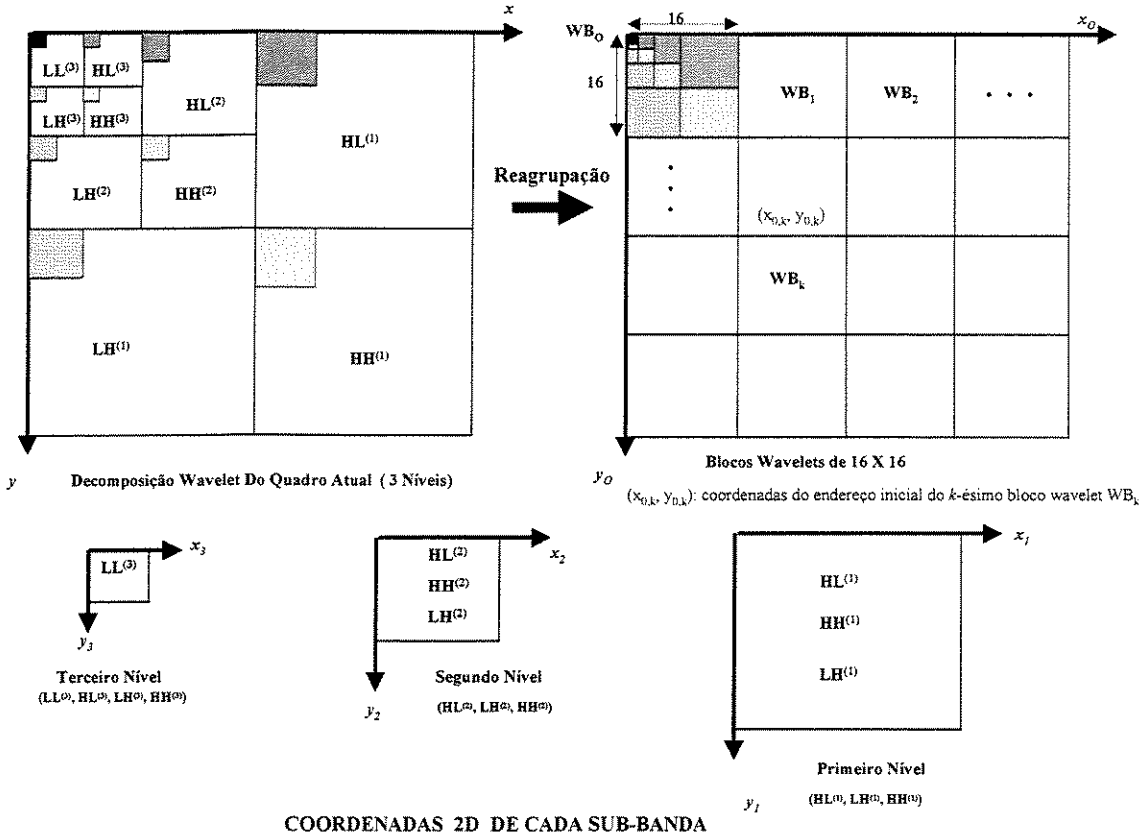


Fig.6.2: Reorganização dos coeficientes *Wavelets* (3 níveis de decomposição) em blocos *Wavelets*.

Esse algoritmo tem como objetivo minimizar uma medida de disparidade entre o "bloco *"wavelet* atual" e o "bloco *wavelet"* de referência". Para cada bloco *wavelet* realiza-se uma busca no quadro de referência pela região escolhida ("janela de busca") que proporcione o melhor casamento, ou seja, que minimize uma medida de disparidade (distância métrica) entre os dois blocos. A Fig. 6.3 apresenta um exemplo geral de como isso ocorre. Nessa figura, o deslocamento do bloco é dado pelo vetor $\vec{d}$, que representa a diferença entre a posição do bloco do quadro atual de referência (que propicia o melhor casamento) e a posição do bloco atual.

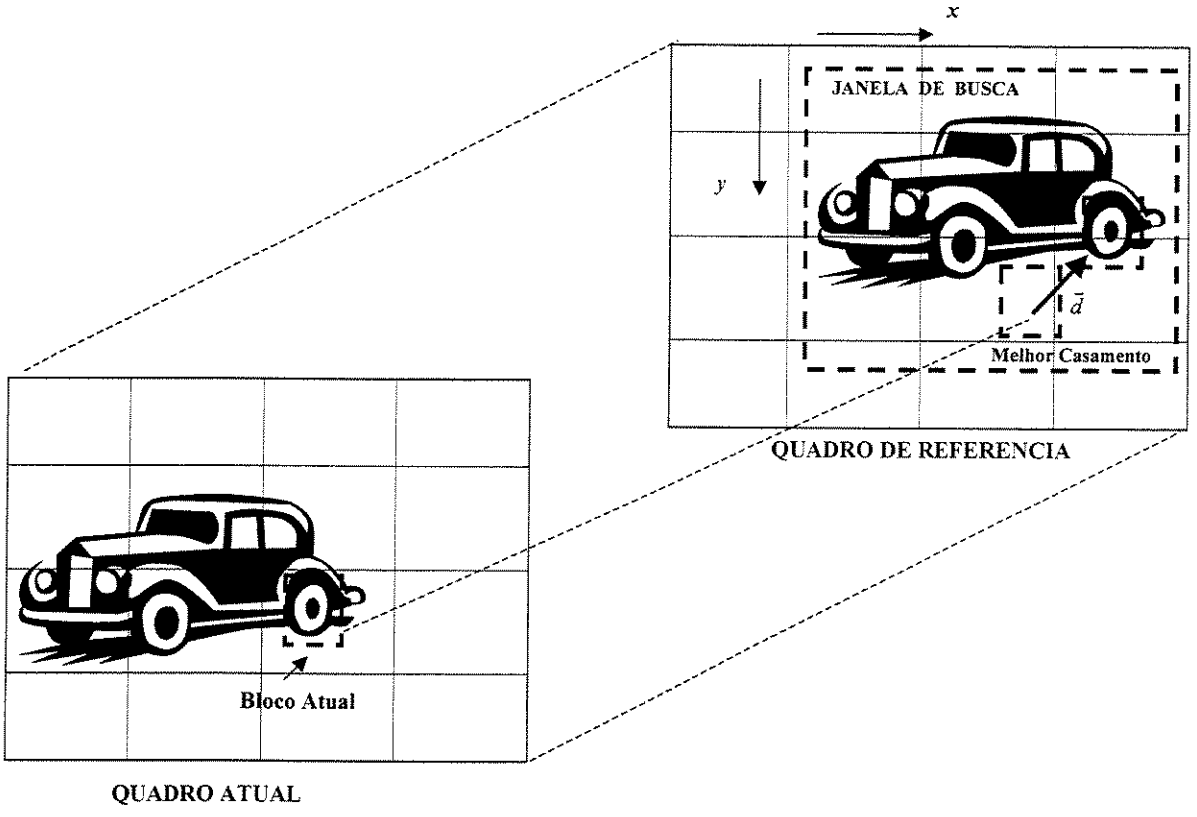


Fig.6.3: Estimação de movimento.

Duas medidas de disparidade muito utilizadas [84, 85, 86, 87, 96, 97] são: o erro quadrático médio (MSE-*"Mean Square Error"*) definida pela Eq.6.3, e a média da diferença absoluta entre pixels (MAD-*"Mean Absolute Difference"*), definida pela Eq.6.4:

$$MSE(i,j) = \frac{1}{M.N} \sum_{m=a}^{M+a} \sum_{n=b}^{N+b} \left[Q_a(m,n) - Q_{ref}(m+i, n+j) \right]^2 \qquad (6.3)$$

$$MAD(i,j)=\frac{1}{M.N}\sum_{m=a}^{M+aN+b}\sum_{n=b}\left|Q_a(m,n)-Q_{ref}(m+i,n+j)\right| \tag{6.4}$$

onde $Q_a(x,y)$ é o quadro atual,

$Q_{ref}(x,y)$ é o quadro de referência

(i, j) é o deslocamento entre os dois blocos $\left(\vec{d}=(i,j)\acute{o}timo\right)$,

(a, b) é a posição do bloco atual.

Entre as duas medidas apresentadas, o MAD tem a menor complexidade computacional. Por outro lado, se um sistema de compressão de vídeo digital tem como objetivo minimizar o MSE para uma determinada taxa de bits, pode ser vantajosa a utilização do MSE como figura de mérito na etapa de estimação de movimento.

Se um sinal de entrada (imagem) é decomposto em três níveis, então a imagem de entrada, ficará decomposta num total de 10 sub-bandas: três do primeiro nível, três do segundo nível e quatro do terceiro nível.

Nesse caso, o vetor deslocamento é $\vec{d}=\left(d_x,d_y\right)$ e o MAD do k- ésimo bloco *wavelet*, WB_k da Fig. 6.2, pode ser calculado pela seguinte expressão:

$$\begin{aligned}
&MAD_k(d_x,d_y)= \\
&=\sum_{i=1}^{3}\sum_{x_3=x_{i,k}}^{x_{i,k}+M/2^i-1}\sum_{y_i=y_{i,k}}^{y_{i,k}+N/2^i-1}\left\{\left|HL_{cur}^{(i)}(x_i,y_i)-HL_{ref}^{(i)}\left(dx\%2^i,dy\%2^i,x_i+\left\lfloor\frac{d_x}{2^i}\right\rfloor,y_i+\left\lfloor\frac{dy}{2^i}\right\rfloor\right)\right|+\right. \\
&+\left|LH_{cur}^{(i)}(x_i,y_i)-LH_{ref}^{(i)}\left(dx\%2^i,dy\%2^i,x_i+\left\lfloor\frac{d_x}{2^i}\right\rfloor,y_i+\left\lfloor\frac{dy}{2^i}\right\rfloor\right)\right|+ \\
&\left.+\left|HH_{cur}^{(i)}(x_i,y_i)-HH_{ref}^{(i)}\left(dx\%2^i,dy\%2^i,x_i+\left\lfloor\frac{d_x}{2^i}\right\rfloor,y_i+\left\lfloor\frac{dy}{2^i}\right\rfloor\right)\right|\right\}+ \\
&+\sum_{x_3=x_{3,k}}^{x_{3,k}+M/2^3-1}\sum_{y_3=y_{3,k}}^{y_{3,k}+N/2^3-1}\left\{\left|LL_{cur}^{(3)}(x_3,y_3)-LL_{ref}^{(3)}\left(dx\%2^3,dy\%2^3,x_i+\left\lfloor\frac{d_x}{2^3}\right\rfloor,y_3+\left\lfloor\frac{dy}{2^3}\right\rfloor\right)\right|\right\}
\end{aligned} \tag{6.5}$$

onde as coordenadas do ponto inicial $(x_{i,k},y_{i,k})$ do i-ésimo nível de sub-bandas no k-ésimo bloco *wavelet* são definidas como:

$$x_{i,k}=\frac{x_{o,k}}{2^i},\ y_{i,k}=\frac{y_{o,k}}{2^i}\ \text{para}\ i=1,2,3.$$

As coordenadas $(x_{0,k}, y_{0,k})$, do k-ésimo bloco no domínio espacial são mostradas na Fig. 6.2, sendo que:

$HL_{ref}^{(i)}$ é a sub-banda *HL ("high - low")* do *i*- ésimo nível do quadro de referência,

$LH_{ref}^{(i)}$ é a sub-banda *LH ("low - high")* do *i*- ésimo nível do quadro de referência,

$HH_{ref}^{(i)}$ é a sub-banda *HH ("high - high")* do *i*- ésimo nível do quadro de referência,

$LL_{ref}^{(3)}$ é a sub-banda *LL ("low - low")* do terceiro nível do quadro de referência,

$M \times N$ é o tamanho dos blocos *wavelets*, nas simulações deste trabalho e é de 16 x 16.

Da Eq. (6.5) pode-se obter o vetor de movimento ótimo $(dx_{\min}, dy_{\min})$, o qual apresenta o erro mínimo de deslocamento,

$$(dx_{\min}, dy_{\min})_k = \min_{(dx,\, dy) \in W} MAD_k(dx, dy) \tag{6.6}$$

onde W é a janela de busca para a estimação de movimento.

Deve-se ressaltar que vários pesquisadores [84, 85, 96], usam a estimação de movimento com exatidão de "meio-*pixel*" (*half - pixel accuracy*) a fim de se reduzir o erro de compensação de movimento. Entretanto, a estimação e compensação de movimento com exatidão de "meio-*pixel*" são dificultosas no domínio *wavelet*. Desde que a TWD é variante com o deslocamento, o processo de interpolação para uma exatidão de "meio-*pixel*" pode gerar no domínio *wavelet* um erro variante com o deslocamento. Uma possível solução é gerar uma imagem com resolução de "meio-*pixel*" no domínio espacial e aplicar a técnica do LSB. No entanto, esse processo requer uma alta carga computacional.

III.3. Estimação e Compensação de Movimento Banda a Banda utilizando-se o Método do Deslocamento da Banda Baixa

Utilizando-se a característica de similaridade entre a subbanda atual e a correspondente subbanda de referência, um eficiente método de estimação de movimento pode ser realizado. Combinando-se a técnica do LSB com a estimação de movimento banda a banda, escapa-se do problema da variação com o deslocamento da TW. Isso permite aperfeiçoar a eficiência do codificador de vídeo no domínio *wavelet*. O tamanho do bloco para estimação e compensação, depende da subbanda, sendo que nesse caso corresponde a um tamanho de 16 x 16, no domínio espacial. A estimação de movimento utilizando-se LBS pode ser aplicada na codificação de vídeo de alta resolução.

Seja $V_{k,\,i,\,B}$ o vetor de movimento do subbloco B do i-ésimo nível, no k-ésimo bloco *wavelet* . Tem-se:

$$V_{k,i,B} = \left(dx_{\min}, dy_{\min}\right)_{k,i,B} = \min_{(dx,dy)\in W} MAD_{k,i,B}(dx,dy) \tag{6.7}$$

Então o MAD desse subbloco B, é dado por:

$$MAD_{K,i,B} = \sum_{x_i = x_{i,k}}^{x_{i,k}+M/2^i-1} \sum_{y_i = y_{i,k}}^{y_{i,k}+N/2^i-1} \left| B_{cur}^{(i)}(x_i, y_i) - B_{ref}^{(i)}\left(dx\%2^i, dy\%2^i, x_i + \left\lfloor \frac{dx}{2^i} \right\rfloor, y_i + \left\lfloor \frac{dy}{2^i} \right\rfloor \right) \right| \tag{6.8}$$

onde B representa as subbandas HL, LH, HH, e LL que conjuntamente com as outras variáveis foram apresentadas na Eq. (6.5).

Quando são aplicados três níveis de decomposição *wavelet*, 10 vetores de movimento são obtidos num bloco *wavelet*. Os 10 vetores para o k-ésimo bloco *wavelet* são $V_{k,1,HL}$, $V_{k,1,LH}$, $V_{k,1,HH}$, $V_{k,2,HL}$, $V_{k,2,LH}$, $V_{k,2,HH}$, $V_{k,3,HL}$, $V_{k,3,LH}$, $V_{k,3,HH}$ e $V_{k,3,LL}$.

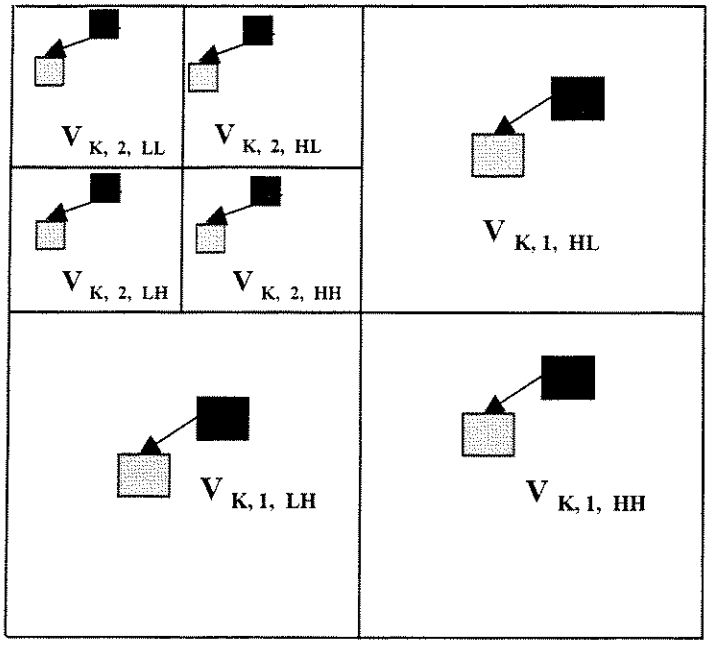


Fig.6.4: Vetores de movimento do método banda a banda para 2 níveis decomposição *Wavelet*.

III.4. "BLOCO *WAVELET*" VERSUS "BANDA A BANDA"

A estimação de movimento banda a banda utilizando-se o método do LSB, difere da estimação de movimento de blocos *wavelet* somente no uso de múltiplos vetores de movimentos para um bloco *wavelet*, como é mostrado na Fig. (6.4). O número de vetores de movimentos para um bloco *wavelet* é igual ao número de sub-bandas no caso em que se

aplica o método de estimação de movimento banda a banda, enquanto que a estimação de movimentos baseada em blocos *wavelet*, necessita de apenas um único vetor de movimento para cada bloco *wavelet*. Porém, no caso da estimação de movimento banda a banda múltiplos vetores de movimentos para cada bloco *wavelet* devem ser codificados. A taxa de bits para esses múltiplos vetores de movimentos é insignificante quando se faz a comparação com a taxa de bits do sinal residual da compensação de movimentos na codificação de vídeo de alta resolução. Por conseqüinte, o desempenho de um codificador baseado na estimação de movimento banda a banda pode ser superior ao de um baseado na estimação de movimento de blocos *wavelet*, especialmente na codificação de vídeo de alta resolução.

Na Fig 6.5 mostra-se a visão hierárquica, do diagrama geral da estimação de movimento para ambos os métodos de estimação de movimento.

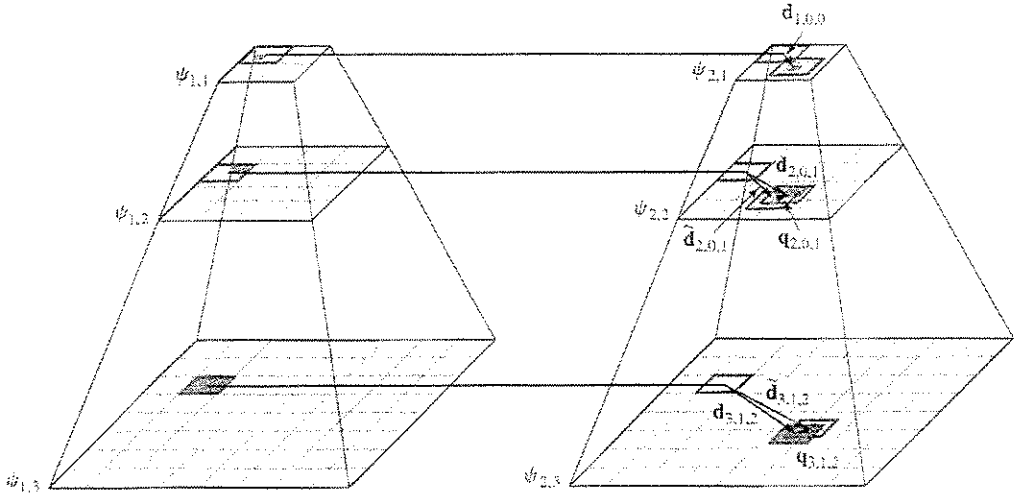


Fig.6.5.- Estimação de movimento hierárquica com estrutura piramidal.

IV. CODIFICADOR .

Na Fig. 6.6 mostra-se um diagrama em blocos do codificador proposto baseado na TW utilizando-se a técnica de estimação e compensação de movimento proposta na Secção III do presente capítulo. No esquema proposto, o sinal de vídeo de entrada é decomposto a partir da aplicação da TW. No bloco onde é realizado o deslocamento da banda baixa, são geradas três bandas *LL* (*low-low*) as quais são deslocamentos de 1 pixel da banda *LL* do quadro de referência ao longo das direções "*x*", "*y*" e diagonal. Para todos os níveis de decomposição da imagem, as operações de deslocamento e de transformação *wavelet* da

banda *LL* são iterativamente realizadas. O processo de estimação encontra um bloco de referência que tem o melhor casamento com o bloco *wavelet* atual. O bloco *wavelet* é constituído pelos coeficientes *wavelets*, os quais são gerados no bloco "deslocamento da banda baixa" da Fig.6.6. O sinal residual proveniente da compensação é o sinal diferença entre o bloco *wavelet* atual e o correspondente bloco *wavelet* de referência. O sinal residual é quantizado e codificado com o codificador *zerotree* descrito na Seção 4.1. Os vetores de movimento são comprimidos num codificador aritmético.

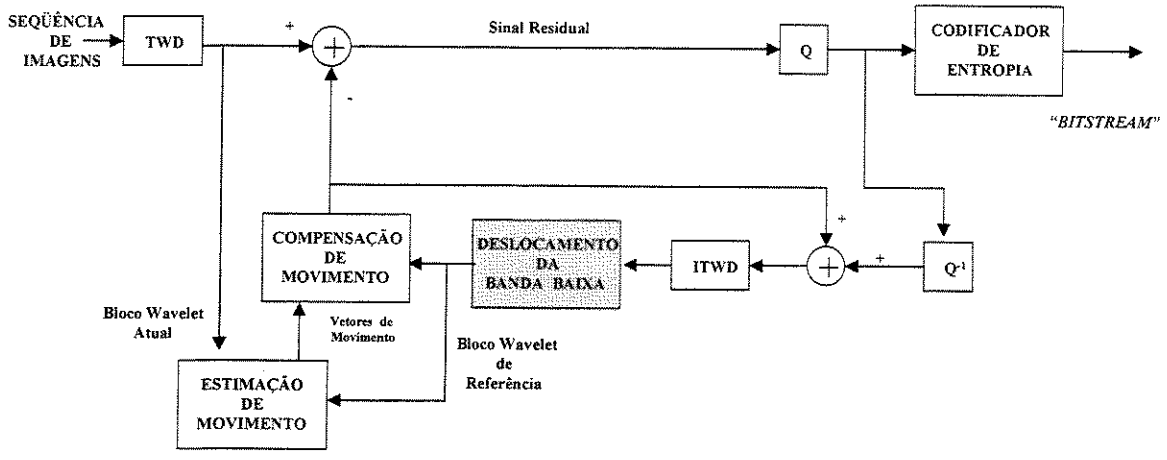


Fig.6.6- Diagrama em blocos de um codificador de vídeo que utiliza os métodos de estimação e compensação propostos.

CAPÍTULO 7.

SIMULAÇÕES E RESULTADOS.

I. INTRODUÇÃO

Neste Capítulo mostram-se os testes e resultados parciais e totais realizados em cada um dos sistemas, assim como os fundamentos que levaram a proposta do esquema final onde se propõe uma técnica para a estimação de movimento no domínio *wavelet* a partir da propriedade da variância com o deslocamento, juntamente com o modelo de pré-processamento conhecido como PACC.

A avaliação foi realizada a partir de três testes. Primeiro foram realizados testes a fim de se comprovar a eficiência isolada do esquema do subsistema codificador discutido no Cap. 4 (Teste 1). A seguir, foram feitos os testes para se avaliar a modelo PACC, no pré-processamento do sinal de vídeo utilizando-se técnicas baseadas na transformada *wavelet*, como foi apresentado no Cap.5 (Teste 2). Finalmente, avalia-se o método proposto.

II. TESTES DO SUBSISTEMA CODIFICADOR DE IMAGEM BASEADO NA TRANSFORMADA *WAVELET* E DA TRANSMISSÃO PROGRESSIVA.

Avaliou-se o desempenho do sistema codificador de imagem baseado na TW, proposto no Cap.4 que é utilizado no codificador de vídeo. As imagens de teste utilizadas foram: Lena, Rose, Winter e Berries. Todas elas com dimensão 256 x 256. Essas imagens são apresentadas nas figuras, Fig. 7.1, Fig.7.2, Fig.7.3 e Fig.7.4.

Para se avaliar o desempenho desse sistema utilizou-se a medida objetiva: relação sinal-ruído de pico (PSNR- *Peak Signal to Noise Relation*), dada em decibéis por :

$$PSNR(dB) = 10\log_{10}\frac{M^2}{MSE} \tag{7.1}$$

onde,

M é o maior valor pico a pico que o sinal de entrada pode assumir, tipicamente 255 para imagens em 8 bits.

MSE é o erro quadrático médio (*mean square error*) entre a imagem e a imagem original sendo calculado por:

$$MSE = \frac{1}{VH}\sum_{i=0}^{V-1}\sum_{j=0}^{H-1}\left(x(i,j) - x_f(i,j)\right)^2 \tag{7.2}$$

onde $x(i,j)$ é a amostra da imagem original e $x_f(i,j)$ é a amostra reconstruída, ambas de dimensões VxH.

Fig. 7. 1: Lena

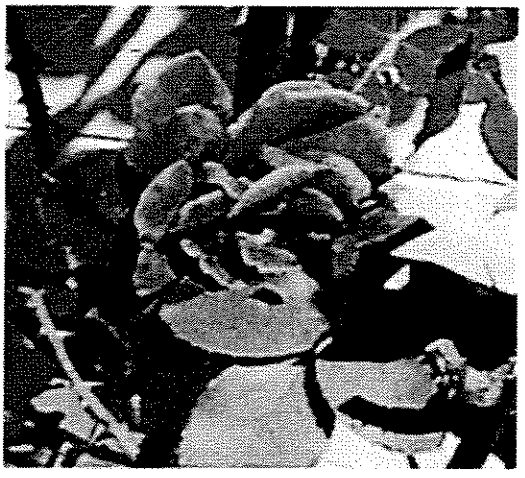


Fig. 7. 2: Rose

Fig. 7. 3: Winter

Fig. 7. 4: Berries

As imagens são decompostas a partir da transformada *wavelet* diádica, utilizando-se filtros: Haar, Daubechies (ordem 4 e 6), Spline 9/7. Os valores dos coeficientes são arredondados para o inteiro mais próximo. Os resultados então, são codificados em planos

de bits, de acordo com o método determinado pelo algoritmo, conforme foi descrito nas seções anteriores.

As figuras, Fig. 7.5 e Fig. 7.6 mostram as medidas "relação sinal–ruído de pico", PSNR e a "**%** pixel erro", respectivamente, como uma função do número de planos de bits para a imagem Lena. Essa "**%** pixel erro", é medida a partir da imagem diferença obtida da subtração da imagem decodificada da original. Note-se que esse algoritmo de codificação-decodificação pode ser considerado quase sem-perdas, quando todos os planos de bits são incorporados. Esse é um dos resultados esperados, desde que o algoritmo tenta codificar uma representação binária completa dos coeficientes transformados, sem jogar fora qualquer informação. Porém, um pequeno erro de quantização é introduzido já que os coeficientes *wavelets* são valores em ponto flutuante e o algoritmo somente decodifica a parte inteira desses valores. Também são introduzidos pequenos erros numéricos quando são aplicadas as operações de cálculo da TW e a inversa da TW. Esse é o motivo pelo qual aparece um valor 0,99% de pixel erro de remanescência (Fig. 7.6), com 38,7dB de PSNR (Fig. 7.5), quando todos os planos de bits são decodificados (plano de bit 14).

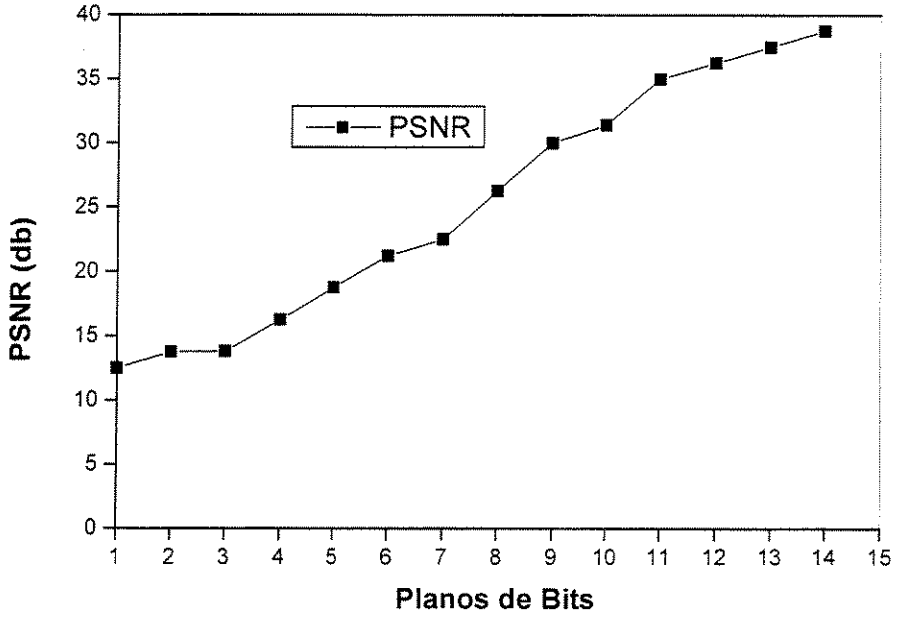


Fig.7. 5: PSNR versus planos de bits (Imagem Lena).

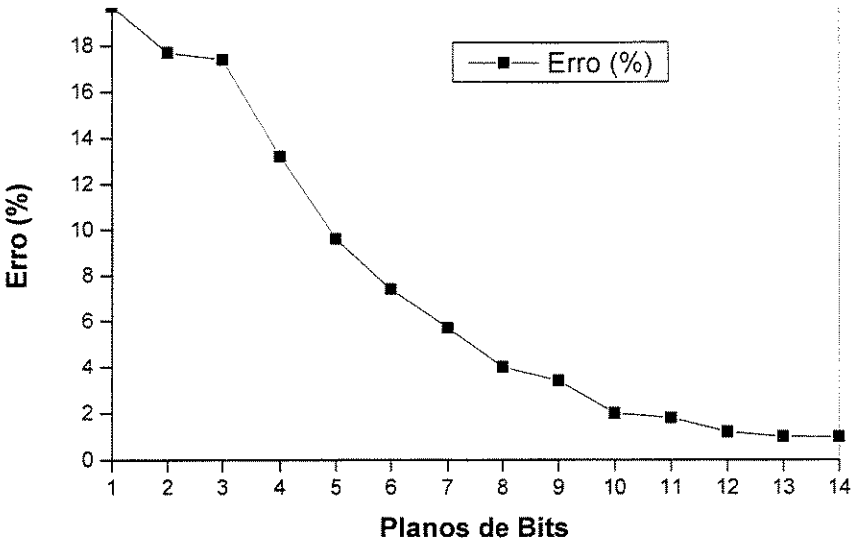


Fig.7. 6: Pixel erro versus planos de bits (Imagem Lena).

As figuras, Fig.7.7 e Fig.7.8 mostram de formas diferentes, o comportamento da taxa de compressão como uma função dos planos de bits. No algoritmo implementado, à medida que se utilizam menos planos de bits, a taxa de compressão aumenta. Note-se que com a eliminação de alguns planos de bits, é possível obter-se uma boa qualidade da imagem acompanhada de uma modesta taxa de compressão. Por exemplo, para o plano de bit 10, a imagem decodificada tem 2 % de pixel erro (Fig. 7.6), com 31,4dB de PSNR (Fig. 7.5), e uma taxa de compressão de 1.6 bits por pixel o que é equivalente a 5:1 (Fig. 7.8). Por esse motivo é que a quantidade de planos de bits é incluído no cabeçalho de informações do codificador.

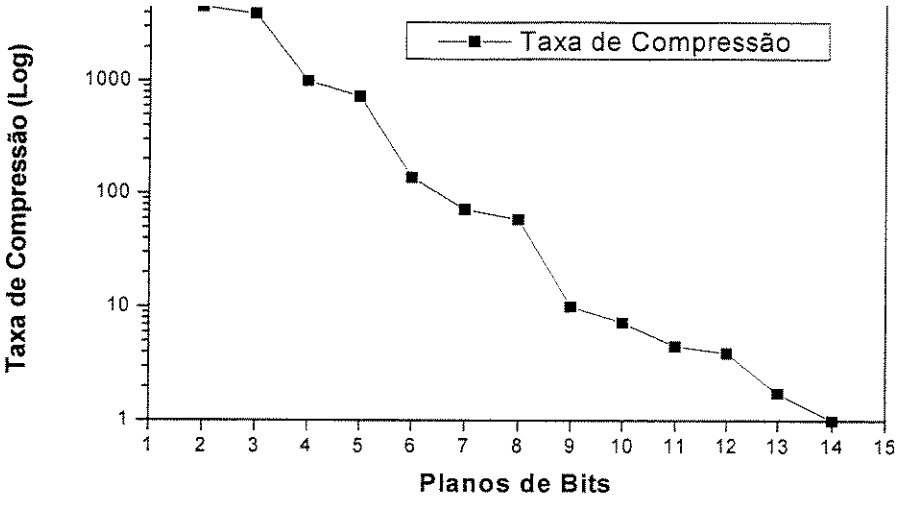


Fig.7.7: Taxa de compressão versus planos de bits (Imagem Lena).

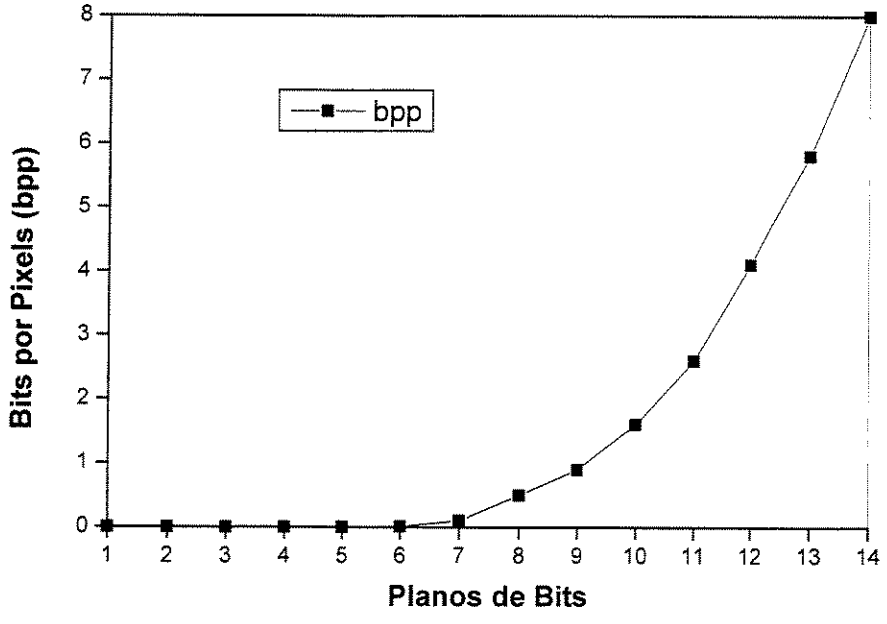


Fig.7. 8: Taxa (bits por pixels) versus planos de bits. Imagem Lena.

A curva da Fig.7.9 permite avaliar o desempenho do algoritmo do ponto de vista objetivo através de "taxas de distorção" onde a taxa é medida em bits por pixel e a qualidade em PSNR. Essa medida fornece uma idéia da degradação da qualidade da imagem recuperada.

Os gráficos mostrados nas figuras, Fig.7.10 e Fig.7.11 apresentam a dizimação como uma forma adicional de se aumentar a taxa de compressão do algoritmo. A dizimação é aplicada nos coeficientes *wavelets* antes da codificação *zerotree*. Note-se que em todos os planos em que são conservados os coeficientes, os resultados são aproximadamente iguais para o

algoritmo de dizimação básico em termos de imagem erro. As figuras Fig.7.10 e Fig.7.11 realizam uma comparação entre o algoritmo de dizimação básico e o algoritmo *zerotree* implementado, com 10 planos de bits, e aplicando-se TW *Daubechies* (N=4) na imagem Lena. Observe-se que para cerca de 10 % de coeficientes *wavelets* retidos, a taxa de compressão da dizimação é de 2,5:1 (Fig.7.10) e a PSNR de 31dB (Fig.7.11). Quando se aplica *zerotree*, a taxa de compressão é de 7,6:1 (Fig.7.10) e a PSNR de 30,5dB (Fig.7.11). Assim, o desempenho do *zerotree* utilizando-se dizimação é superior em termos de compressão.

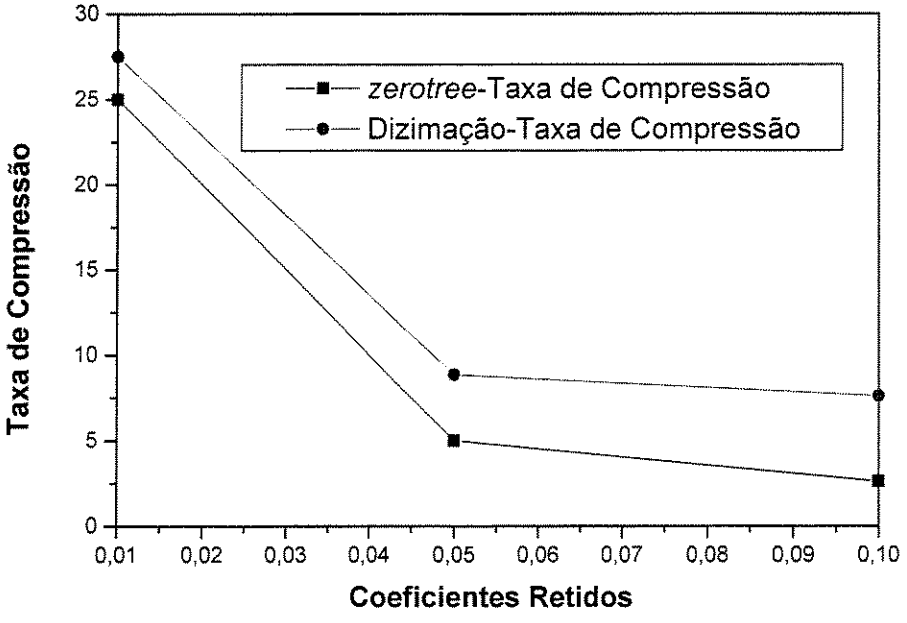


Fig.7. 10: Taxa de compressão versus coeficientes retidos (Imagem Lena para 10 planos de bits).

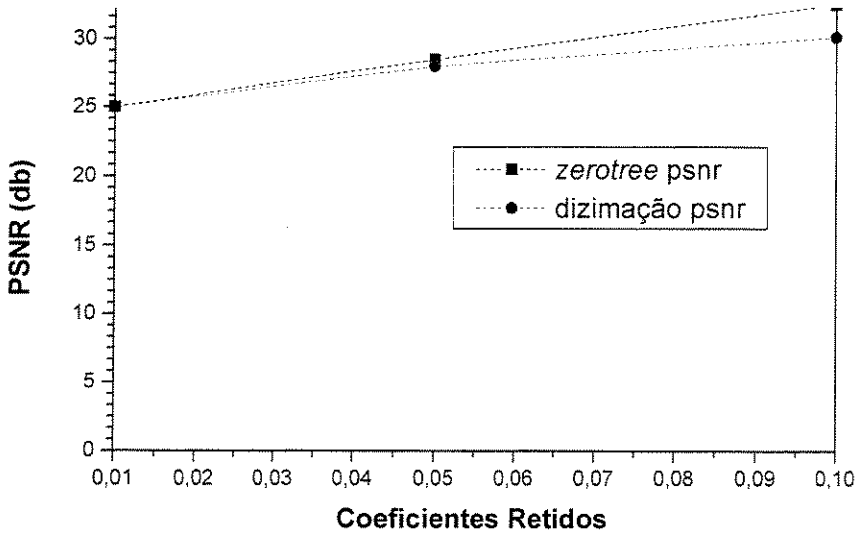
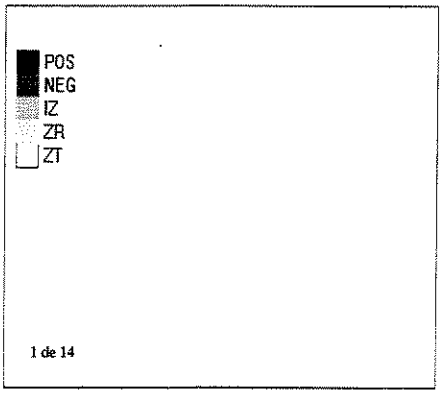
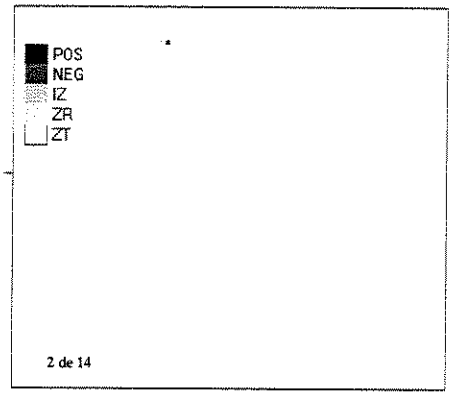


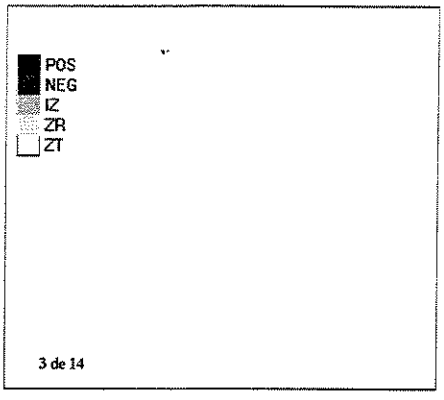
Fig.7. 11: PSNR versus coeficientes retidos (Imagem Lena para 10 planos de bits).

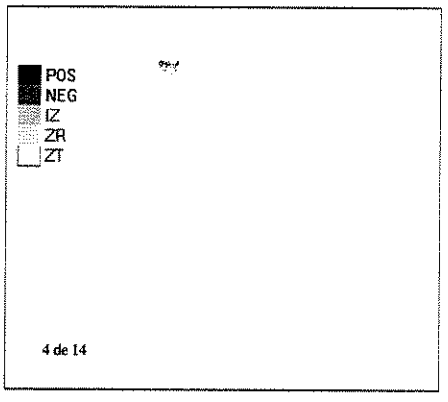
Nas figuras Fig.7.12 e Fig.7.13 mostra-se a técnica de transmissão progressiva. Esses planos de bits da Fig.7.12 são correspondentes a codificação *zerotree* da imagem Lena, utilizando-se *wavelet* Daubechies de ordem 6. Nas imagens mostradas, os quatro símbolos codificados e escritos no arquivo de saída : **POS**, **NEG**, **IZ**, **ZR**, são representados usando-se tonalidades que variam do cinza ao branco, sendo que o símbolo **ZT** é representado na cor branca. Pode-se observar, que nos planos de 1 até 7 são predominantemente brancos, o que significa que esses planos de bits, não requerem uma alta capacidade de armazenamento. De fato, os primeiros três planos de bits combinados utilizam menos de 30 bytes, e a combinação dos sete primeiros precisam, somente de aproximadamente de 1 kb para ser armazenado. Essa característica é muito útil e deve ser explorada no esquema de compressão, uma vez que os primeiros planos, contém os bits mais significativos, e conseqüentemente, uma informação significativa da imagem. Nos últimos planos de bits, o 13 e 14, acontece o contrário, já que contém os bits menos significativos, e podem ser ignorados no esquema de compressão.

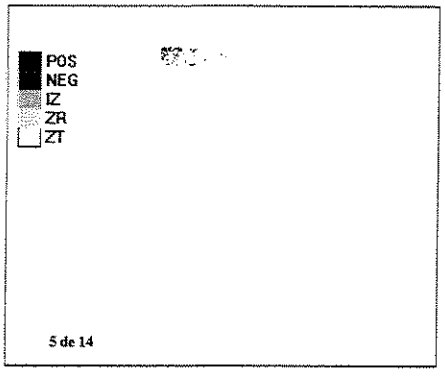
Na Fig. 7.13 mostra-se a seqüência de planos de bits obtida no processo de decodificação da imagem Lena, do plano mais significativo até o menos significativo, mostrando-se a decodificação *zerotree* junto com a técnica de transmissão progressiva. Pode-se observar quanta informação é transmitida, aplicando-se a transformada inversa depois que cada plano de bits é construído.

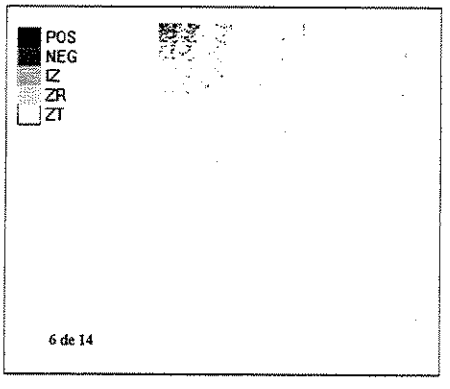

POS
NEG
IZ
ZR
ZT
1 de 14

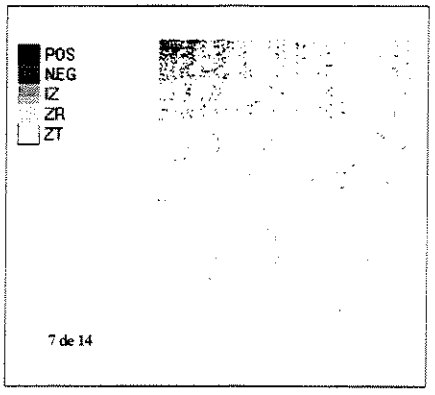

POS
NEG
IZ
ZR
ZT
2 de 14

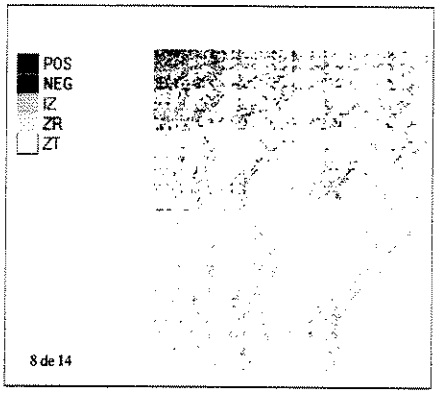

POS
NEG
IZ
ZR
ZT
3 de 14

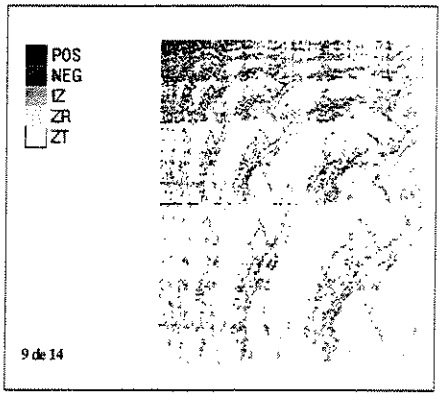

POS
NEG
IZ
ZR
ZT
4 de 14

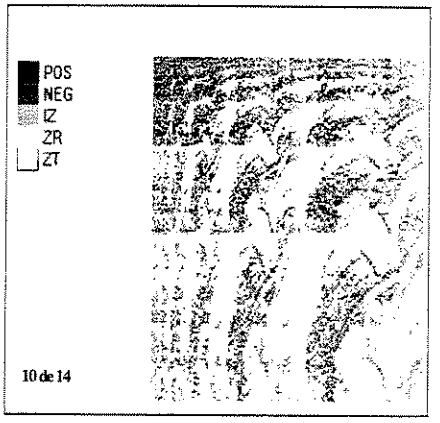

POS
NEG
IZ
ZR
ZT
5 de 14

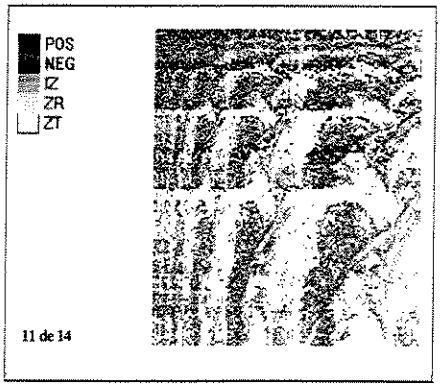

POS
NEG
IZ
ZR
ZT
6 de 14

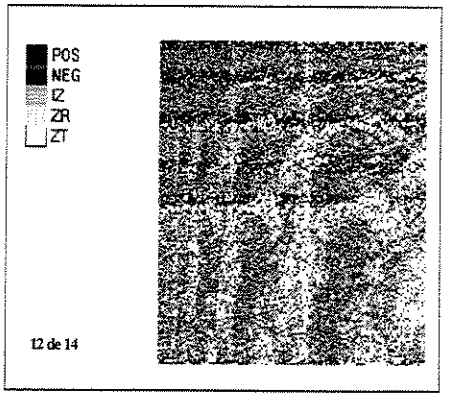


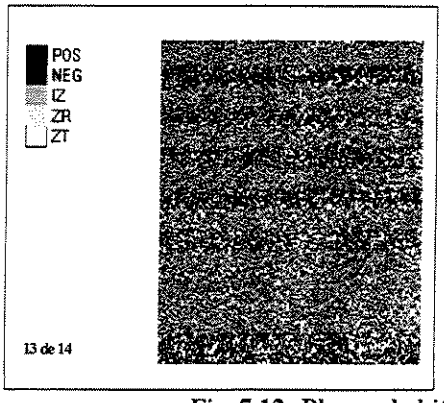












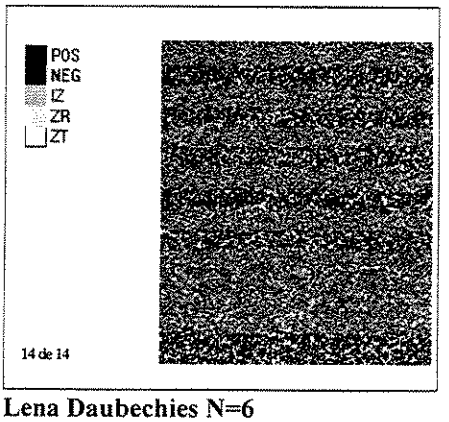
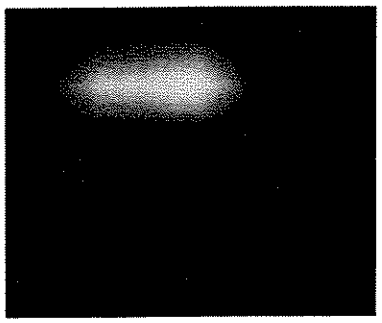
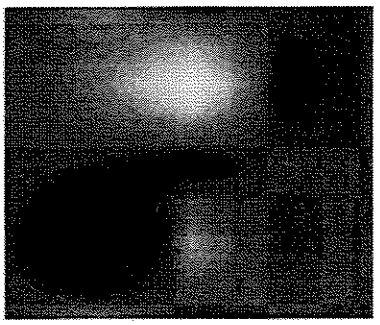


Fig. 7.12: Planos de bits. Codificação da Imagem Lena Daubechies N=6



1 de 14



2 de 14

3 de 14

4 de 14

5 de 14

6 de 14

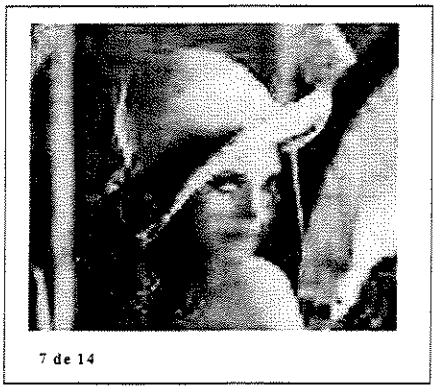


Fig. 7.13. Planos de bits. Decodificação progressiva da Imagem Lena. (correspondente a Fig. 7.12).

II.1. TESTES COMPLEMENTARES

O objetivo desses testes é o de avaliar a seleção dos diferentes tipos de *wavelets*, que serão utilizados no algoritmo. Os testes foram realizados tomando-se como referência os trabalhos realizados por Mandal et al. [98] e Villasenor et al. [99]. Mandal, Panchanthan & Aboulnasr (1996), discutem e propõem um método que permite selecionar o tipo de *wavelets* a partir da quantificação do conteúdo de detalhes de uma imagem. Também Villasenor, Belzer & Liano (1995), propõem uma forma de seleção, do tipo de *wavelets* a partir da avaliação de um total de 4300 tipos de *wavelets* [100].

Na Tabela 7.1 mostra-se que uma imagem ou segmento de imagem que se caracterizam por apresentar uma grande quantidade de regiões homogêneas, ou uma tonalidade constante ou linearmente variável, como é o caso das imagens Lena e Rose, a TW de *Daubechies* tem um melhor desempenho em relação ao erro de compressão que a TW de *Haar*. Porém para a imagem Winter que contem mais detalhes de informação de alta freqüência do que as imagens Lena e Rose, o erro de compressão é aproximadamente o mesmo para os três tipos de *wavelets* (Ver Tabela 7.2).

Tabela 7. 1: Pixel Erro(%) da imagem Lena utilizando-se 10%, 5% e 1% dos coeficientes das TW.

TW	10 % Coeficientes	5 %Coeficientes	1 %Coeficientes
Haar	2,0376%	2,8925%	5,1009%
***Daubechies* (N=4)**	1,7066%	2,5006%	4,6811%
***Daubechies* (N=6)**	1,6699%	2,4530%	4,7459%

Tabela 7. 2: Pixel Erro(%) da imagem Winter utilizando-se 10%, 5% e 1% dos coeficientes das TW.

TW	10 % Coeficientes	5 %Coeficientes	1 %Coeficientes
Haar	4,5452%	5,7742%	8,1284%
***Daubechies* (N=4)**	4,4540%	5,6570%	6,8949%
***Daubechies* (N=6)**	4,4371%	5,6506%	6,8417%

III. TESTE DA TÉCNICA PACC NO PRÉ-PROCESSAMENTO DO SINAL DE VÍDEO.

O objetivo principal desses testes é comprovar a interoperabilidade do conjunto de algoritmos, pré- codificação, escalonabilidade e a compensação local de movimentos bem como, técnicas e ferramentas que foram projetados de forma independente, em condições operacionais diferentes, e agora formarão parte de um sistema único (codificador), de

acordo com o que foi visto no Cap.4. As seqüências de vídeo utilizadas nas simulações são: Kiel, Mobile, e Tennis. O primeiro quadro de cada uma dessas seqüências é apresentado nas Figs. 7.14, 7.15 e 7.16. A dimensão dos quadros é de 720 x 480 e a taxa de quadros é de 30 quadros/s. Essas seqüências apresentam características que possibilitam representar bem o universo de seqüências reais. Dentre essas características, pode-se citar a presença de objetos com deslocamentos lineares e circulares, *zoom* (muito marcante na seqüência Kiel) e movimentos panorâmicos de câmera (*pan*) [83].

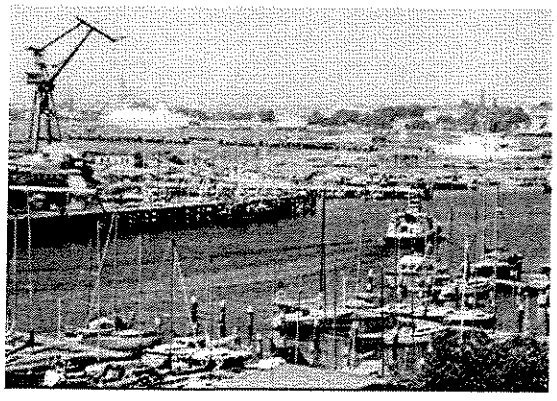


Fig.7.14: Kiel

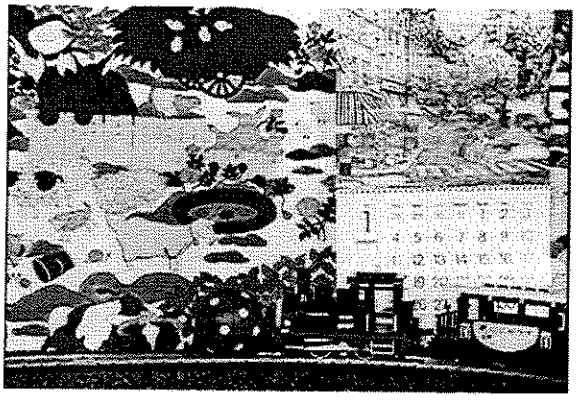


Fig.7.15: Mobile

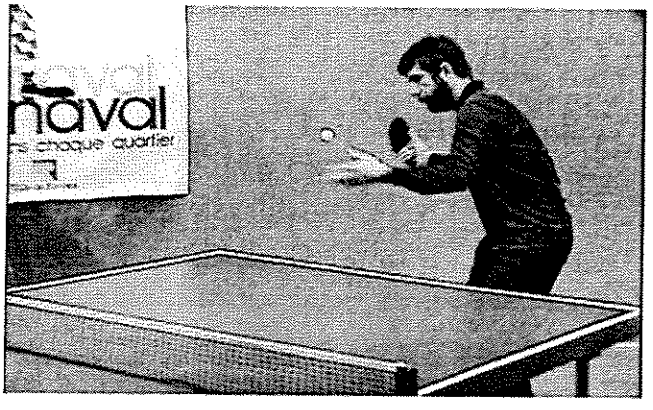


Fig.7.16: Tennis

A fim de se avaliar o desempenho do sistema foram obtidas as curvas de PSNR(dB) versus taxa de distorção em bits por pixel, onde, considera-se o RMSE médio (considerando-se as componentes de cor R, G e B), calculado como:

$$RMSE = \sqrt{\frac{MSE_R + MSE_G + MSE_B}{3}} \tag{7.3}$$

Os resultados das simulações são apresentados nos gráficos das Figs. 7.17, 7.18 e 7.19. O sistema foi analisado comparando-se os resultados obtidos nas simulações do modo hierárquico (escalonamento SNR), com o caso em que não há escalonamento de dados, ou seja, toda a informação é enviada por um único canal. Numa segunda comparação foi considerado, que no modo hierárquico, existem somente os dados da camada principal.

Nas simulações realizadas, foram utilizados os erros quadráticos médio, como figura de mérito, e a busca em três passos, como estratégia de busca, usando-se vetor de deslocamento. O tamanho da janela de busca foi de [- 15,15]. A ordenação dos coeficientes é feita utilizando-se varredura alternada especificada no padrão MPEG-2 [101]. A codificação é sempre realizada no modo progressivo. Os passos de quantização são determinados pelas matrizes de quantização, do Padrão MPEG-2 [68].

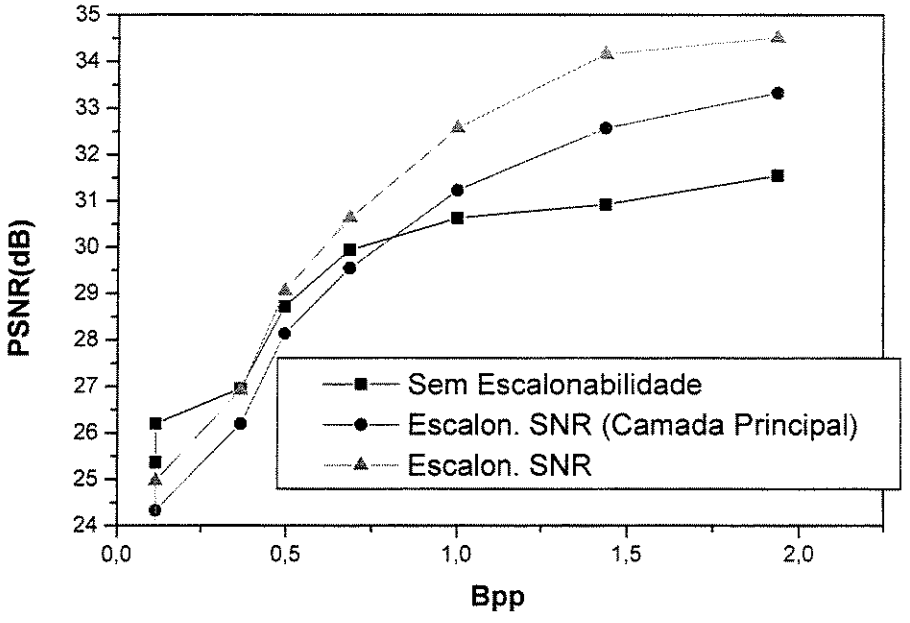


Fig.7.17: Avaliação do desempenho do sistema para a seqüência Tennis.

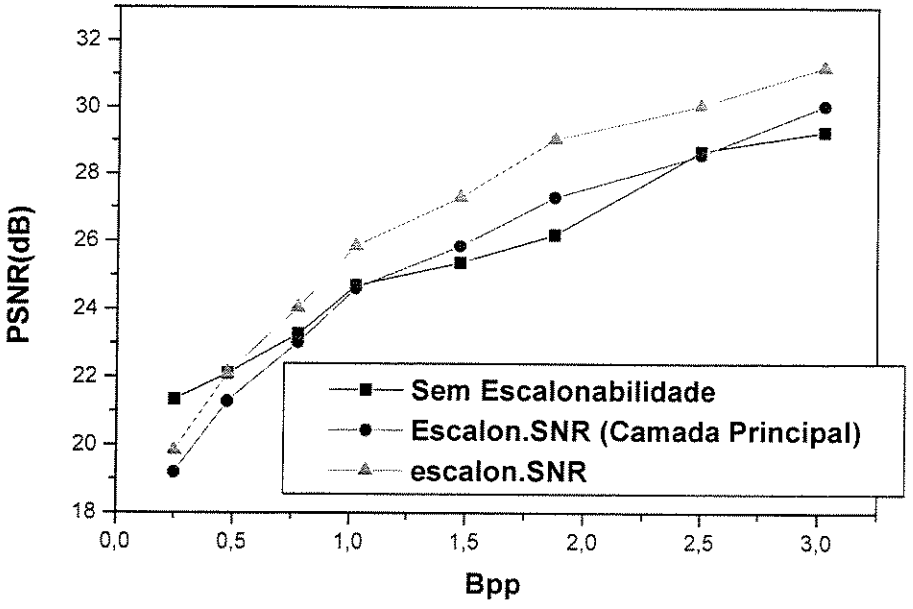


Fig.7.18: Avaliação do desempenho do sistema para a seqüência Mobile.

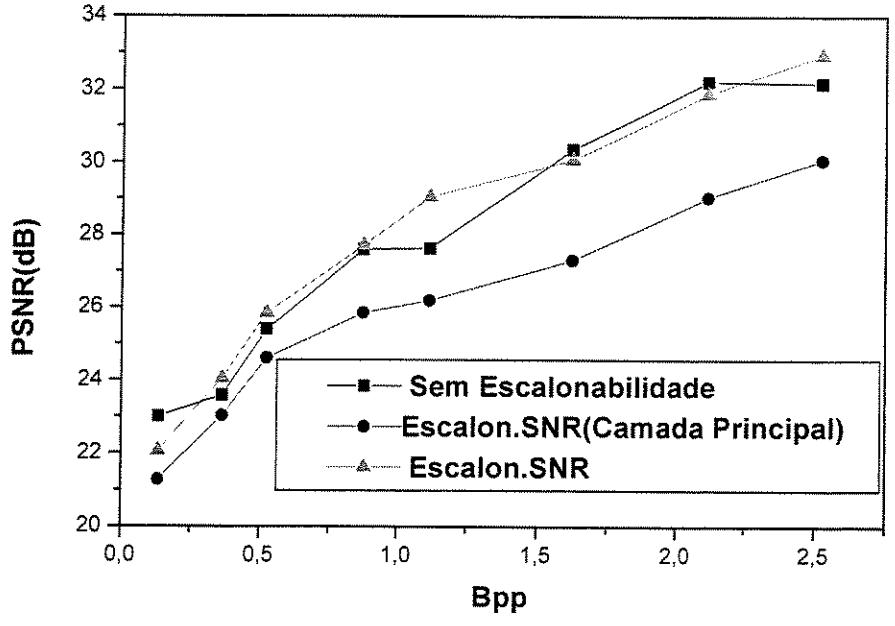


Fig.7.19: Avaliação do desempenho do sistema para a seqüência Kiel.

As simulações realizadas mostram a compatibilidade funcional apresentada pelo conjunto de técnicas implementadas e combinadas nesse codificador. Através desses gráficos nota-se

o bom desempenho do sistema, em cada modo, ainda com o uso da escalonabilidade na situação em que não existe o canal de baixa prioridade, especialmente quando só a camada principal é utilizada na etapa de predição. Entretanto, também é importante considerar o bom desempenho quando o sistema opera em condições normais, ou seja, quando o canal de baixa prioridade existe. Analisando-se os gráficos obtidos fica evidente a vantagem de se utilizar alguma técnica de escalonabilidade, junto com o PACC, para taxas superiores a 0,5 bits/pixel.

III. Testes do Esquema Final. Método Proposto.

O esquema com o método proposto e duas técnicas convencionais de estimação de movimentos, para a codificação de imagens de vídeo, baseados na transformada *wavelet* foram comparados. O primeiro dos métodos tradicionais foi utilizado para se avaliar a estimação de movimento com a técnica de casamento de blocos, com blocos de tamanho de 16 x 16 no domínio espacial [91], e decompor o sinal residual da compensação do movimento em três níveis de transformação *wavelet*. O segundo método convencional foi utilizado para se observar o desempenho da estimação de movimento, sem o método de deslocamento da banda baixa. Nesse, a estimação de movimento foi realizada diretamente "banda a banda" no domínio *wavelet* [90, 91]. No esquema com o método proposto, a estimação de movimento é realizada com os métodos "banda a banda" e "blocos *wavelet*" utilizando-se a técnica de "deslocamento de banda baixa". Esses métodos são comprados com os métodos convencionais de estimação, mencionados anteriormente, com relação a curvas de distorção versus taxa, sendo a distorção representada em PSNR (dB), e a taxa dada em bits por pixel dentro das condições apresentadas a seguir.

As seqüências utilizadas, foram Kiel, Mobile e Tennis, que foram mostradas no Cap.5. A janela de busca, usada foi de [-16,16]. A estimação foi realizada com casamento de blocos, com blocos de 16 x 16 pixels. Os filtros *wavelets* utilizados foram do tipo Spline biortogonais 9/7, com três níveis de decomposição. O primeiro quadro da seqüência foi utilizado como sendo o quadro de referência. Na compensação de movimento do sinal residual no domínio *wavelet*, foi utilizado o algoritmo proposto no Cap.4.

Nas figuras, Fig.7.20, Fig.7.21, Fig.7.22 e Fig.7.23 apresentam-se, os desempenhos dos algoritmos convencionais e do proposto, a partir da curva de taxa de distorção. A taxa de

bits apresentada na figura inclui, a compressão dos dados, da compensação de movimento do sinal residual e dos vetores de movimento.

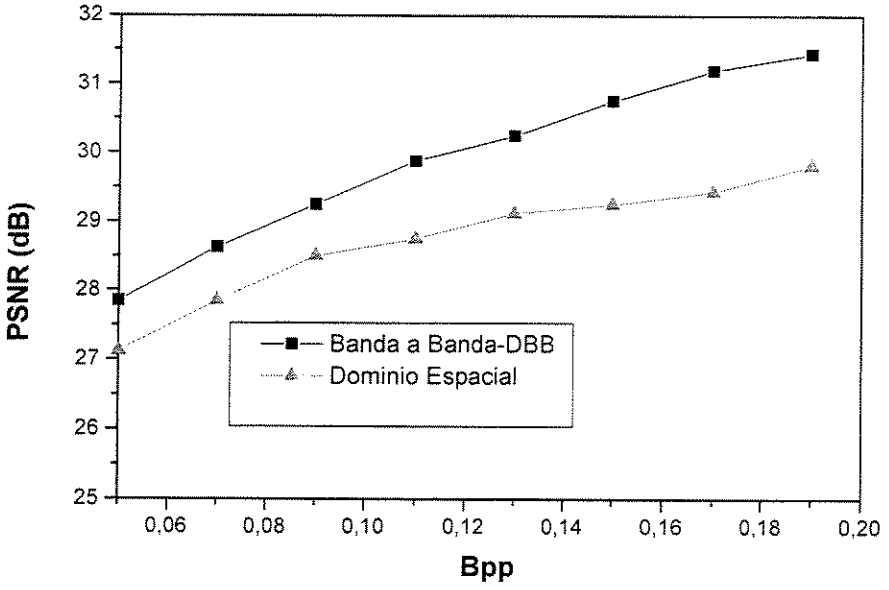
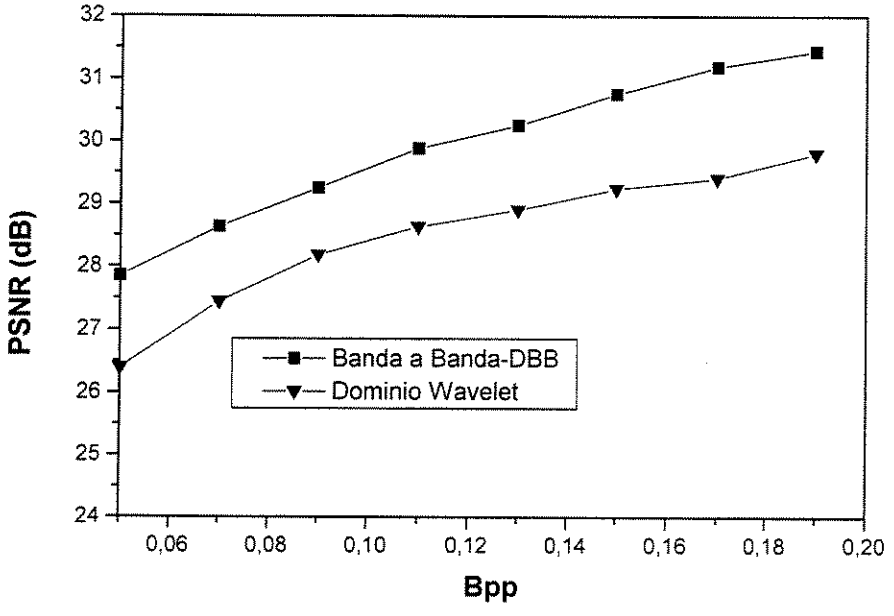


Fig.7.20: Curvas de taxa de distorção obtidas utilizando-se o esquema proposto a fim de se comparar o método Banda a Banda com o método convencional no domínio espacial, para a seqüência Tennis.



.Fig.7.21: Curvas de taxa de distorção obtidas utilizando-se o esquema proposto a fim de se comparar o método Banda a Banda com o método convencional no domínio *wavelet*, para a seqüência Tennis.

Observa-se nas Figs 7.21 e 7.22 em média, um aumento de 1,5dB da PSNR, no método de estimação de movimento banda a banda, proposto, sobre o método de estimação no domínio espacial e *wavelet* convencional.

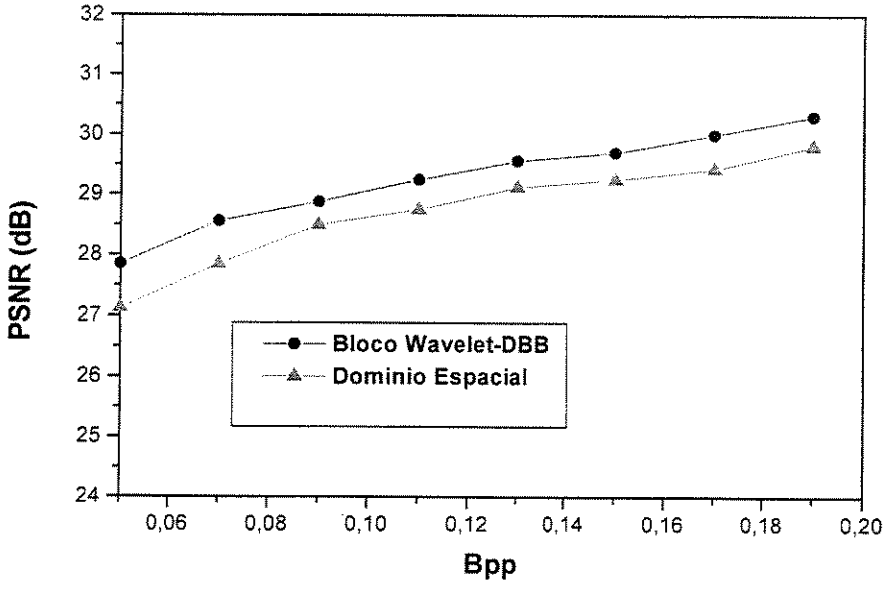


Fig.7.23: Curvas de taxa de distorção obtidas utilizando-se o esquema proposto a fim de se comparar o método Blocos *Wavelet* com o método convencional no domínio espacial, para a seqüência Tennis.

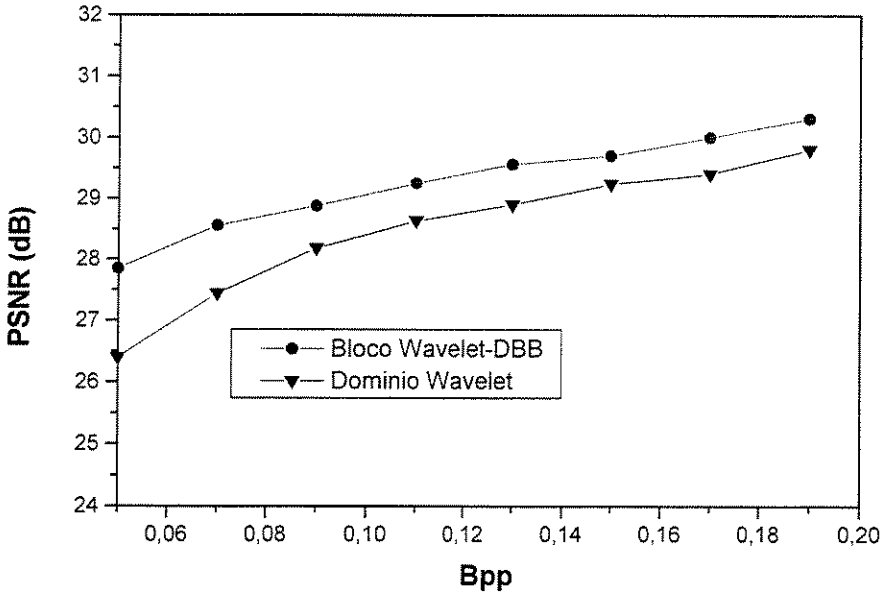


Fig.7.24: Curvas de taxa de distorção obtidas utilizando-se o esquema proposto a fim de se comparar o método Blocos *Wavelet* com o método convencional no domínio espacial, para a seqüência Tennis.

Das Figs. 7.23 e 7.24 pode-se avaliar que o esquema proposto, utilizando-se o método blocos *wavelets* apresenta uma taxa de distorção PSNR que em média é 0.7dB superior aos outros métodos convencionais empregados na avaliação.

Na Tabela 7.1 apresenta-se uma avaliação da complexidade computacional e os requisitos de memórias de cada um dos métodos. A principal desvantagem do esquema proposto é o grande requisito de memória e a complexidade computacional. Por exemplo, para três níveis de decomposição *wavelet*, quando se incorpora o método de deslocamento da banda baixa, um total de nove quadros na memória é requerido, sendo que isso corresponde a três quadros para cada nível *wavelet*. Isso faz com que a complexidade computacional do método proposto seja maior que do que a da estimação de movimento no domínio espacial. A complexidade computacional é calculada pressupondo-se que a estimação de movimento com busca total (*full-search-FS*) seja feita numa imagem de 720 x 480 pixels com uma faixa de busca de ±16. A complexidade computacional do esquema prosposto é 10,3% maior do que a da estimação de movimento no domínio espacial.

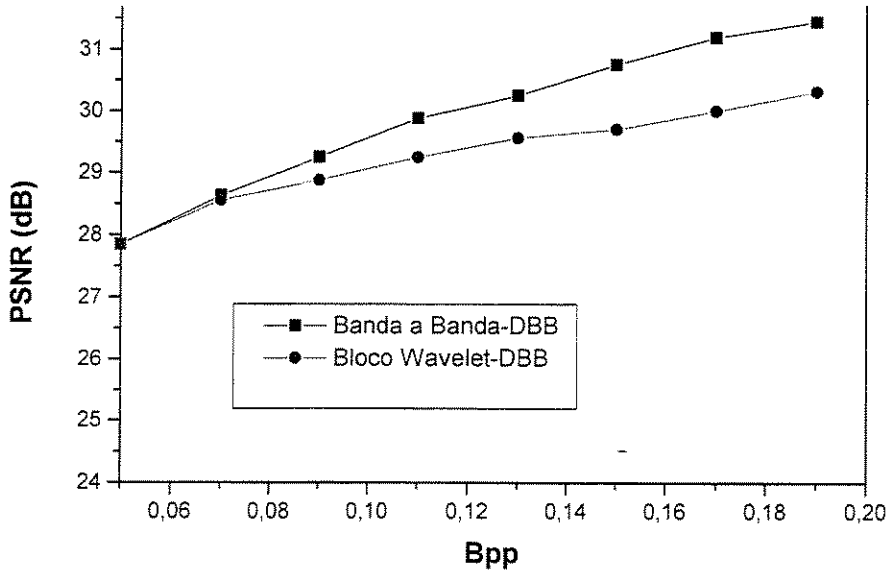


Fig.7.25: Desempenho do Método proposto para a seqüência Tennis.

Tabela 7.3: Complexidade Computacional e Requerimentos de Memória.

	DOMÍNIO ESPACIAL	DOMÍNIO WAVELET	ESQUEMA PROPOSTO
REQUERIMENTO DE MEMÓRIA	1 quadro	1 quadro	10 quadros
COMPLEXIDADE COMPUTACIONAL	FS+TW+ITW 1.1 Giga operações	TW+FS 1.1 Giga operações	TW+ITW+LBS+FS 1.2 Giga operações

Dessa forma, sugere-se o método de estimação de blocos *wavelets*, para ser aplicado em codificadores de baixa taxa de bits, enquanto que o método de banda a banda pode ser aplicado eficientemente para aplicações de alta resolução, desde que a quantidade de bits comprimida, para os múltiplos vetores de movimento no método banda a banda não seja desprezível num codificador de baixa taxa de bits. O esquema proposto tem como objetivo central evitar ou fugir do problema que representa para as técnicas convencionais de compensação e estimação de movimentos a propriedade da variação com o deslocamento da TWD. O método de deslocamento da banda baixa, pode ser aplicado nos codificadores de vídeo no domínio *wavelet*.

CAPÍTULO 8.

8.1. Conclusões

8.1.1. SOBRE AS POTENCIALIDADES DA ANÁLISE *WAVELET*

No estudo realizado comprovou-se, que:

1. A localização espaço-freqüência;
2. A compactação de energia e a similaridade entre subbandas em diferentes escalas;
3. O decaimento da magnitude dos coeficientes *wavelets* entre subbandas à medida que se refina a escala de resolução;
4. O *Clustering* de coeficientes significativos em uma subbanda;

são as cinco propriedades estatísticas da transformada *wavelet* que se apresentam como bases das estratégias inovadoras na organização e representação dos dados de uma imagem transformada.

Dentre os algoritmos *wavelets,* considerados como "estado da arte" em compressão de imagens, pode-se selecionar quatro algoritmos que representam os melhores da lista de codificadores *wavelets* para imagens, sendo eles:

- EZW -"*Embeded Zerotree Wavelet Coder*"
- SPIHT - "*Set Partitioning in Hierarchical Trees*"
- MRWD - "*Morphological Representation of Wavelet Data*"

Todos eles exploram eficientemente as propriedades estatísticas dos coeficientes *wavelets* resultantes da decomposição da pirâmide diádica.

Esses codificadores quando são comparados com codificadores baseados em blocos da imagem, como o JPEG, apresentam um desempenho superior em relação a qualidade visual, devido principalmente à ausência do efeito de blocos.

Por outro lado, em relação ao desempenho objetivo, a partir da PSNR, esses algoritmos apresentam um aumento da ordem de 1-3 dB sobre os codificadores por transformada em bloco.

EZW é hoje um referencial, dentro dos compressores de imagem tanto em eficiência quanto em simplicidade computacional. De fato, muitos algoritmos, entre eles o LZC o SPIHT, foram desenvolvidos a partir do EZW.

O SPIHT (o algoritmo que mais se destacou como uma evolução significativa do EZW) apresenta alto desempenho em termos de taxa de distorção e simplicidade computacional. Explora mais eficientemente a similaridade entre as subbandas quando comparado ao EZW, o que conduz a um aumento significativo na PSNR de 0,86-0,94dB sobre o EZW.

Demonstrou-se também que em regiões de textura e bordas o uso da predição fractal é mais eficiente, porém um método híbrido *wavelet* -fractal pode ser aplicado para se melhorar a eficiência da codificação.

8.1.2. Sobre os Testes Preliminares

O objetivo central dos testes realizados é o de avaliar o desempenho de algoritmos e técnicas *wavelets* na compressão de imagens estáticas, visando a aplicação desses resultados no processamento de sinais de vídeo.

Teste 1: Subsistema Codificador de Imagem

O resultados obtidos mostram bom desempenho e eficiência no esquema que realiza a codificação *zerotree* quando usado em conjunto com a técnica de transmissão progressiva, uma vez que os primeiros planos de bits não requerem alta capacidade de armazenamento. De fato, os primeiros três planos de bits combinados utilizam menos de 30 bytes, e a combinação dos sete primeiros necessitam aproximadamente de apenas 1 Kb para ser armazenado.

Essa característica é muito útil e pode também ser explorada em outros esquemas de compressão, uma vez que os primeiros planos contém os bits mais significativos, e conseqüentemente uma informação significativa da imagem. Nos últimos planos, que contém os bits menos significativos, estes podem ser ignorados no esquema de compressão.

Comprovou-se também o alto nível de informação transmitida aplicando-se a transformada inversa depois que cada plano de bits é construído.

Os resultados obtidos através de "% pixel erro", mostram que o algoritmo de codificação-decodificação pode ser considerado quase sem-perdas, quando todos os planos de bits são incorporados.

Comprovou-se que a eliminação de alguns planos de bits possibilita obter boa qualidade da imagem acompanhada conseqüentemente de uma taxa de compressão mais baixa.

Foi apresentada a dizimação como forma adicional de se aumentar a taxa de compressão do algoritmo, comprovando-se assim, que o desempenho do algoritmo *zerotree* utilizando dizimação é superior em termos de compressão.

Teste 2: Técnica PACC no pré-processamento do sinal de vídeo e na codificação hierárquica.

Comprovou-se a compatibilidade do conjunto de algoritmos presentes no MPEG-2, na pré-codificação no domínio *wavelet* utilizando-se a técnica PACC, na escalonabilidade e na compensação local de movimentos.

A proposta revela que o processo de codificação do sinal de vídeo baseada na transformada *wavelet* com compensação local de movimentos se torna mais eficiente quando se realiza pré-processamento ou pré-codificação do sinal, a partir de técnicas que exploram as redundâncias das estruturas *wavelets,* como é o modelo PACC, que envolve os conceitos de codificação condicional, partição e agregação.

Obteve-se bom desempenho quando o sistema opera em condições normais, ou seja, quando existe o canal de baixa prioridade.

Analisando-se os resultados, observa-se a vantagem de se utilizar alguma técnica de escalonabilidade em conjunto com o PACC, para taxas superiores a 0,5 bits/pixel.

O teste mostrou a boa compatibilidade funcional que resulta do conjunto de técnicas implementadas e combinadas nesse codificador.

8.1.3. Sobre o Método Proposto

No esquema contendo o método proposto, a estimação de movimento é realizada com os métodos "banda a banda" e "blocos *wavelet*" utilizando-se a técnica de "deslocamento de banda baixa".

A partir dos resultados, sugere-se a utilização do método de estimação de blocos *wavelets*, para ser aplicado em codificadores de baixa taxa de bits, enquanto o método de banda a banda pode ser aplicado eficientemente para aplicações de alta resolução, desde que a quantidade de bits comprimida, para os múltiplos vetores de movimento no método banda a banda não seja desprezível num codificador de baixa taxa de bits.

Devido à propriedade da variância com deslocamento que é uma propriedade da TWD, os resultados propostos podem ser estendidos para qualquer banco de filtros *wavelet*.

Demonstrou-se que a exploração da propriedade de similaridade, entre as subbandas *wavelets* atuais e a correspondente subbanda de referência, permite aperfeiçoar a eficiência do codificador no domínio *wavelet*.

O esquema proposto, utilizando-se o método de blocos *wavelets*, a taxa de distorção PSNR é em média 0.7dB superior quando comparada com os métodos convencionais empregados.

O método de estimação de movimento banda a banda, proposto, em média, tem aumento de 1,5dB na PSNR sobre o método de estimação no domínio espacial usando-se *wavelet* convencional.

O esquema proposto tem como objetivo central evitar ou fugir do problema que representa para as técnicas convencionais de compensação e estimação de movimentos a propriedade da variação com o deslocamento da TWD. O método de deslocamento da banda baixa pode ser aplicado nos codificadores de vídeo no domínio *wavelet*.

8.1.4. Sobre as Limitações da Proposta.

A principal desvantagem do esquema proposto é o grande requisito de memória e a complexidade computacional. Por exemplo, para três níveis de decomposição *wavelet*, quando se incorpora o método de deslocamento da banda baixa, se requer um total de nove quadros na memória, sendo que isso corresponde a três quadros para cada nível *wavelet*. Isso faz com que a complexidade computacional do método proposto seja maior do que a do método de estimação de movimento no domínio espacial convencional.

8.2. Sugestões para Trabalhos Futuros

Atualmente, apesar de todos os resultados advindos da teoria fractal e implementação de codificadores, a curva de desempenho taxa-distorção de um codificador fractal dificilmente atinge o desempenho do estado da arte em codificadores *wavelets* tais como EZW, SPIHT, MRWD, SLCCA, descritos no Cap.4 deste trabalho. Por esse motivo adota-se, atualmente uma abordagem híbrida *wavelet*- fractal, onde com um custo computacional adicional, pode-se melhorar o desempenho de um codificador *wavelet*.

Seguindo-se a mesma linha deste trabalho pode-se propor um codificador de vídeo combinando-se algoritmos híbridos *wavelet*- fractal.

Esse esquema pode ser construído a partir dos algoritmos de processamento de imagens estáticas, que dentro desta linha, estão mostrando uma evolução significativa, sendo eles:

PPC -"*Predictive Pyramid Coder*"

SQS - "*Self-Quantization of Subtrees*"

WFC - "*Wavelet-Fractal Coder*"

BIBLIOGRAFIA

[1] Kenneth R. C. Digital Image Processing. USA, 1998. 388 p.

[2] Grossman A., Morlet J. Decomposition of Hardy Funtion into Square Integrable Wavelet of Constante Shape. SIAM J. Math., USA, v.15, pp. 723-736, 1984.

[3] Chui C. K. An Introdution to Wavelet. Academic Press, San Diego, 1992.

[4] Daubechies I. Ten Lectures on Wavelets. Society for Industrial and Applied Mathematics, Philadelphia, 1992.

[5] Young M. R. An Introdution to Nonharmonic Fourier Series. Academic Press Inc. , Ohio, 1980.

[6] Chui C. K. An Introdution to Wavelets. Academic Press Inc. , San Diego, 1992.

[7] Young R. K. Wavelet Theory and Aplications. Kluwer Academic Publihers, USA ,1993.

[8] Vetterli M. , Herly C. Wavelets and Filter Bank: Theory and Design. IEEE Transactions on the Signal Processing, USA,v.40, n. 9, p. 2207 - 2232, Setembro, 1992.

[9] Vetterli M. , Kovacevic J. Wavelet and Subbanda Coding. New Jersy. Prentice - Hall, 1995. 488p.

[10] Oppenheim A.V. , Schafer R.W. Digital Signal Processing. Englewood Cliffs Prentice Hall Inc., 1975.

[11] Porat B. Digital Processing of Random Signals. Prentice Hall, Pennsylvania, 1992.

[12] Mallat S. A Wavelet Tour of Signal Processing. Academic Press, San Diego, 1997.

[13] Mallat Stephane G. A Theory for Multiresolution Signal Descomposition: the Wavelet Representation. IEEE Transactions on Pattern Analysis and Machine Intelligence, USA, v.11, n.7, Julho 1989

[14] Adelson E. H. , Simoncelli E. , Hingorani R. Orthogonal Pyramid Transforms for Image Coding. Processding of SPIE, v.845, pp. 50 - 58, Cambrige, Outubro, 1987.

[15] Mallat S. Multiresolution Approximation and Wavelets. IEEE Transactions on Pattern Analysis and Machine Intelligence, v.11, n.7, USA, Julho, 1989.

[16] Daubechies I. Orthogonal Bases of Compactly Supported Wavelets. Comunications on Pure and Applied Mathematics, v.XLI, p.909-996, USA, 1988.

[17] Strang G. Wavelets and Dilatation Equations. SIAM Review, v.31, p. 613-627, 1989.

[18] Rao Raghuveer M. , Bopardikar A. S. Wavelets Transform: Introduction to Theory and Applications. Addison Wesley, California, 1998,pp. 308.

[19] Mallat S. Multifrequency Channel Decompositions of Images and Wavelet Models. IEEE Transactions on Acustics, Speech and Signal Processing, v.37, n.12, Julho, 1989.

[20] Founier A. Wavelets Algorithms and Applications in Compurer Graphics SIGGRAPH'94. Course Notes

[21] Esteban D. , Galand Applications of Quadrature Mirror Filters to Split band Voice Coding Schemes. Proc. Int. Conf. Acoust. Speech Signal, Maio, 1977.

[22] Vaidayanathan P. Multirate Systems and Signal Processing. Prestice Hall, Englewoods Cliffs, NJ,1993.

[23] Oppenheim A. V. , Shafer R. W. Discrete-Time Signal Processing. Prestice Hall, Englewoods Cliffs, NJ,1993.

[24] Antonini M., Barland M. , Matheiu P. Duabechies I. Image Coding Using Wavelet Transform. IEEE Transactions on Image Processing, USA, v. 1 n.2, p. 205-220 ,Abril 1992.

[25] Akansu A., Haddad R. Multiresolution signal Decomposition. Academic Press, 1993.

[26] Akansu A., Smith M. Subband and Wavelet Transform. Kluwers, 1995.

[27] Strang G. , Nguyen Wavelets and Filter Banks. Wellesly-Cambridge Press, Boston 1996.

[28] Cohen A. , Daubechies I. , Feauveau J. Biorthogonal bases of Compactly supported Wavelets. Commun. on Pure and Appl. Math., USA, v. 45, p.485-560,1992.

[29] Cohen A. Ondelettes, Analyses Multiresolutions et Filtres miroirs em quadrature. Ann. Inst. H. Poincaré, Anal. Non Linéarie, v.7, p.439-459, 1990.

[30] Vitterli M., Herley C. Wavelets and Filter Banks: Relationships and new result Proc. IEEE ICASSP, Albuquerque, Abril, 1990.

[31] Burt P. , Adelson E. The Laplacian Piramid as a compact image code IEEE Trans. Commun., USA, vol. 31, p. 482-540, 1983.

[32] Daubechies I. Orthogonal Bases of Compactly Supported Wavelets II-Variations on a Theme". ATT&Bell Lab., Tech. Report TM 11217-89111617, USA, 1990.

[33] Meyer Y. Ondelettes et fonctions spline Sem Equations aux Derivates, Partielles, Ecole Polytechnique, Paris, Dezembro 1996.

[34] Meyer Y. Principe d' incertitude, bases hilbertiennes et algebres d' operateurs Bourbaki seminar ,n.662 1995-1996.Strang G. "Wavelet Transform versus Fourier Transforms". Bulletin of the Americam Mathematical Society. USA, v. 26, n. 2, p. 288-304, Abril 1993.

[35] Huffman C. J. Wavelet and Image Compression . SMPTE Journal, USA, Novembro, 1994.

[36] Strang G. , Troung N. Wavelets and Filter Banks. Wellesley Cambridge, 1991, pp520.

[37] Rabbini M., Jones P. W. Digital Image Compressao Techniques. SPIE Optical Engeneering Press, 1991.

[38] Jain K.A. Fundamental of Digital Image Processing. Pretice Hall,Englewood Cliffs, NJ 1989.pp.569.

[39] Brislawn M. C. , Bradly J. N. , Hopper T.The FBI Compression Standard for Digitized Fingerprint Images. Proceeding of The SPIE Conference, pp.2847, Applications of Digital Image Processing XIX 1996.

[40] Shapiro J. M.Embedded Image Coding Using Zerotrees of Wavelets coefficients. IEEE Trans. Signal Processing, USA, v.41, n. 12 p.3445-3462, Dezembro, 1993.

[41] Servetto S. , Ramchandran K. , Orchad M. T. .Wavelet based Image Coding via Morfological Predition of Significance. IEEE Trans. Image Processing, USA, v.41, n. 12 p.3445-3462, Dezembro, 1998.

[42] Said A. , Pearlman W. A.A New Fast and Efficient Image Codec based on set Partitioning in Hierarchical Trees. IEEE Trans. Circuit Syst. Video Technol., USA, v.6, n3 p.243-250, Junho, 1996.

[43] Bing-Bing Ch., Vass J., Zhuang X. Significance-Linked Connect Component analysis for Wavelets Image Coding. IEEE Trans. Image Processing, USA, v.8, n. 6 Junho, 1999.

[44] Li J. Kuo J. C. Image Compression with a Hibrid Wavelet - Fracta Coder. IEEE Trans. Image Processing, USA, v.8, n. 6, Junho, 1999.

[45] Taubman D. , Zakhor A. Multirate 3-D Subbanda Coding of Video. IEEE Trans. Image Processing, Special issue on image Sequence copressionUSA, v.3 n.5, p572-588, Setembro, 1994.

[46] Lim J.S. Two Dimensional Signal and Image Processing. Pretice Hall,Englewood Cliffs, NJ 1990.pp.695.

[47] Ohta M. , Nogaki S. Hibrid Picture Coding with Wavelet Transform and Overlapped Motion-Compesated Interframe Predition Coding. IEEE Transactions on Signal Processing, USA, v.41, n12, p. 3416-3424, Dezembro, 1993.

[48] Martucci S.A. , Sodagar I. , Chiang T. Zhang Q. Y. A Zero Tree Wavelet Video Coder. IEEE Transactions on Circuit and Systems for Video Tecnology, USA, v.7, n1, p.109-118, Fevereiro, 1997.

[49] Hakell B G, Puri A., Netravali A. Digital Video: An Introdution to MPEG-2. Chapman & Hall, New York 1995.pp.400.

[50] Mitchell L.J., Pennebaker B.W., Fogg C. E. , Le Gall J. D. MPEG Video Compression Standard. Chapman & Hall, New York 1997.pp.464.

[51] Bhaskaran V., Konstantinides K. Image and Video Compressio Standards. Algorithms e Architectures. Kluwer Academic Publishser, Boston, 1997.pp. 452.

[52] Uz M. K. "Multiresolution Systems for Video Coding" PhD thesis, Columbia Unversity,New York, Maio, 1992.

[53] Uz M. K., Vitterli M., LeGall D. A Multiresolution approach to Motion Estimation and Interpolation with aplication to Coding of Digital HDTV Proc. IEEE Int. Symp. Circ. and Syst., New Orleans, p. 1298-1301, Maio, 1990.

[54] Uz M. K., Vitterli M., LeGall D. Interpolative Multiresolution Coding of Advanced Television IEEE Trans. on CAS for Video Technology, Special Issue on Signal Processing for Advanced Television, USA, v.1, n.1 p. 86-99, Março, 1991.

[55] Podilchuk C. I. Low-bit rate subband video coding. Proc. IEEE Int. Conf. On Image Prec., v. 3, p. 280-284, Austin, TX Novembro, 1994.

[56] Vitterli M., Uz M. K. Multiresolution Coding Techniques for Digital Video a review Special Issue on Multidimensional Processing of Video Signals Multidimensional Systems and Signal Processing, USA, v.3 p. 161-187, 1992.

[57] Woods J.W. , Naveen T. A filter based bit allocation sheme for subband compression of HDTV . IEEE Trans. on IP, v.1 p.436-440, Julho 1992

[58] Vetterli M. , Kovacevic J., LeGall J.D. Perfect Reconstrução filter bank for HDTV representation and Coding. Image Communication, v.2 n. 3 p.349-364, Outubro 1990.

[59] Queiroz de Farias M. C. Aplicação da Transformada Wavelets na Compressão de Imagens. Campinas. FEEC, UNICAMP. Tese- Mestrado , Junho, 1998.

[60] Comer M.L. et al. Rate - Scalable Coding using a Zero Tree Wavelet Approach. Proceeding of the Ninth Image and Multidimensional Digital Signal Processing workshop, Belize, p. 162-163Março3-6, 1996.

[61] Cardoso F., Meloni L., Machado M. Arantes D. Codificação por Transformada Wavelet. Campinas. DECOM-FEEC.UNICAMP. Notas do Curso, Agosto 1999

[62] Smith, M. J. T. , Eddins, S. L. Subband Coding of Images with Octave Band Tree Structure. Proceedings IEEE International Conference on Acoustic, Speech and Signal Processing. p. 1382-1385.

[63] Brislaw, C. M. Persrvation of the Subband Symetry in Multirate Signal Coding. IEEE Trans. on Signal Processing. v.43, n.12., Dez. 1995.

[64] Lewis, A., Konwles, G. Image Compression Using the 2-D Wavelet Transform . IEEE Trans. on Image Proc. v.1, n.2,p 244-250. 1992.

[65] Said A. , Pearlman W. A. An Image Multiresolution Representation for Lossless and Lossy Compression. IEEE Trans. on Image Proc., USA, v.5, n9 p.243- 250, Sept., 1996.

[66] Ambranson, N. Information Theory and Coding. McGraw-Hill book Company.

[67] ISSO/IEC. Information Technology-Generic Coding of Moving Picture and Associated Audio Information: Video. ISO/IEC 13818-2, 1995.

[68] Wilson D., Ghanbari M. Optimization of MPEG-2 SNR scalable coding. IEEE Trans. Image Process, v 10 n 8. p. 1435-1438. USA. 1999.

[69] Hsu F. , Hsieh H. , Chen Y. Embedded SNR scalable MPEG-2 video encoder and its associated error resilience decoding procedure. Signal Processing Image Communication, v 14, n "5. p. 397-412. USA. 1999.

[70] Gallat M. , Kossentini F. Efficient scalable DCT-based video coding at low bit rates. IEEE International Conference of Image Processing, v 3. p. 782-786.USA.1999.

[71] Yang L. , Martins F. , Gardos T. "Improving H.263 + scalability performance for very low bit rate Applications ". SPIE Conference on Visual Communications and Image Processing. v 3653, p. 768-779. San Jose. CA.1999.

[72] Kodin L. , Katsaggelos A. "An optimal single pass SNR scalable video coder". IEEE International Conference of Image Processing. v 1, p. 276-280.

[73] Arnold J. , Fracter M. , Wang Y. "Efficient drift-free signal-to-noise ratio scalability". IEEE Trans. Circuits Systems Video Technol. v 10, n 1, p 70-82. USA. 2000.

[74] Sablón Becerra.V. , Iano Y. Processamento e Compressão do Espaço de Sinal de Vídeo aplicado às Imagens de Televisão Convencional e HDTV, utilizando a Transformada de Wavelet. DECOM, FEEC, UNICAMP. Relatório No 1 –FAPESP. Faculdade de Engenharia Elétrica, Universidade Estadual de Campinas. Abril, 1999.

[75] Sablón Becerra.V. , Iano Y. Processamento e Compressão do Espaço de Sinal de Vídeo aplicado às Imagens de Televisão Convencional e HDTV, utilizando a Transformada de Wavelet. DECOM, FEEC, UNICAMP. Relatório No 2 –FAPESP. Faculdade de Engenharia Elétrica, Universidade Estadual de Campinas.Mar,2000.

[76] Sablón Becerra.V. , Iano Y. Processamento e Compressão do Espaço de Sinal de Vídeo aplicado às Imagens de Televisão Convencional e HDTV, utilizando a Transformada de Wavelet. DECOM, FEEC, UNICAMP. Relatório No 3 –FAPESP. Faculdade de Engenharia Elétrica, Universidade Estadual de Campinas.Mar,2001.

[77] Marpe D. , Cycon Hans. Very Low Bit Rate Video Coding using wavelet-based Techniques. Picture Coding Symposium. Berlin.. Alemanha. 1997.

[78] Said A. , Pearlman W. A. An Image Multiresolution Representation for Lossless and Lossy Compression. IEEE Trans. on Image Proc., USA, v.5, n9 p.243- 250, Sept., 1996.

[79] Lewis, A., Konwles, G. Image Compression Using the 2-D Wavelet Transform . IEEE Trans. on Image Proc. v.1, n.2,p 244-250. 1992.

[80] Huang Y. , Dreizen H. , Galatsanos N. Prioritized DCT for Compression and Progressive Transmission Images. IEEE Trans. On Image Proc. v 2, n. 4. p.477-487. USA. Outubro. 1992.

[81] Wang Y. , Salari E. The Post Wavelet Transform Redundancy and its Reduction Techniques for Image Compression. Proc. SPIE. v 2418, p.164-173. 1995.

[82] Byung C. , Beon J. A fast multi-resolution block matching algorithm for motion estimation. Signal Processing: Image Communication. v 15. p.799-810. USA. 2000.

[83] Mendes Rômulo L. Transmissão de Vídeo em Pacotes através de Redes de Faixa Larga. Teses de Mestrado. FEEC-Unicamp. Abril.1998.

[84] Watkinson J. The MPEG handbook. MPEG-1, MPEG-2, MPEG-3. Focal Press, Oxford, 2001,pp. 391.

[85] Bovik A. Hadbook of Image & Video Processing. Academic Press, San Diego, 2000,pp. 891.

[86] Rao R. K. , Yip C. P. The transform and Data Compression Handbook . CRC Press, New York, 2001,pp. 388.

[87] Bernd J. Digital Image Processing . Springer, New York, 2002,pp. 585.

[88] Gabor D. Theory of Comunication. Journal of the IEEE, USA, v.33, p.429-457, 1946.

[89] Park T. K. , Jeon G. J. , Kim S. Interframe Coding using tow-stage variable block - size multiresolution motion estimation and wavelet descomposition. IEEE Trans. Circuits systems Video Technol. USA v.8, Agosto1998, p. 399-410.

[90] Mandal K. M. ,Aboulnasr T. , Panchanathan S. Adaptative Multiresolution Motion Estimation Techniques for Wavelet-based video Coding . SPIE Processing: visual Communication and Image Processing, USA, v.3309, n. 6, Janeiro, 1998, p965-974.

[91] Zhang Y. , Zafar S. Motion-compensated Wavelet Transform Coding for Color Video Compression. IEEE Trans. Circuits Systems Video Technol., USA, v.2, Setembro, 1992, p285-296.

[92] Meyer F. , Averbush A. , Coifman R. Motion Compensation of Wavelet coefficients for Very Low bit rate Video Coding. IEEE International Conference on Image Processing, USA, v.3, Outubro, 1997, p638-641.

[93] Caffario C. , Guaragnella C. , Bellifemine A. , Chimienti R. P. Motion Compensation and Multiresolution Coding. Sinal Processing Image Comunication, USA, v.6, Maio, 1994.

[94] Jozawa H. , Watanabe H. "Video Coding with motion Compensated subband/wavelet descomposition". SPIE Processing, USA, v. 1818, 1992. p.765-774.

[95] Park H. W. , Kim H. "Motion Estimation using low-band shift method for wavelet-based moving-picture coding". IEEE Trans. Image Processing, USA, v.9, Junho, 2000. p. 577-587.

[96] Wang Y. , Ostermann J. , Zhang Y. Viddeo Processing and Comunication. Pretice Hall, New Jersy.USA, 2001. pp.595.

[97] Sayood K. Introduction to Data Compression. Academic Press, San Fransisco, USA, 2000. pp.636.

[98] Mandal, M., Panchanathan, S., Aboulnasr, T. Choice of Wavelets for Image Compression Lecture Notes in Computer Science, v.1133, p. 239-249. Springer-Verlag. 1996.

[99] Villasenor, J., Belzer, B., Liao, J. Wavelet Filter Evaluation for Image Compresión. IEEE Trans. on Image Proc., v.4, n.8, p 1053-1060. Ag. 1995.

[100] Welstead, S. Fractal and Wavelet Image Compression Techiques. University of Central Florida. 1999.

[101] ITU-T Recommendation H.262, Information technology - Generic coding of moving pictures and associated audio information: video. Agosto 1995.

[102] Mallat S. , Zhong S. Characterization of Signals from Multiscale Edges. IEEE Transactions on Pattern Analysis and Machine Intelligence, USA, v.14, n.7, Julho 1992.

[103] Daubechies I. The Wavelets Transform Time -frequency Localization and Signal Analysis. IEEE Transactions Information Theory, v.36, n. 5, p. 961 - 1005, Setembro, 1990.

[104] Mallat S. "A Theory for Multiresolution Signal Decomposition". IEEE Transactions PAMI, v.11, n.674-693, Julho, 1989.

[105] Hamming R. W. Error Detecting and error Correcting Coding. Bell Sys. Tech. J., v.29, p.147-160, Julho, 1989.

[106] Franca C. A. Um Codec para Imagens Paradas baseado na Transformada Discreta de Wavelets e Quantização LinearAdaptativa. Anais do XIV Simpósio Brasileiro de Telecomunicações, v.1, p.67-72, 1996.

[107] Sablatash M., Cooker T. , Kida T. The Coding of Image Sequences by Wavelets, Wavelets Packets and Other Subband coding Schemes. IEEE Transactions on Brocasting , USA, v.40, n4, December, 1994.

[108] Davis G. M. A Wavelet-based analysis of Fractal Image Compression . IEEE Trans. Image Processing, USA, v.7, p141-154, Fevereiro, 1998.

[109] Rinaldo R. , Calvano G. Image Coding by Block Prediction of Multiresolution Subimages. IEEE Trans. Image Processing, USA, v.4, p909-920, Julho, 1995.

[110] Press, W., Teukolsky S., Vetterling, W., Flaneery, B. Numerical Recipes in C. 2da ed., Cambrige University Press. 1992.

[111] Herbert, S. Borland C++. Harness the Power of Borland C++. McGraw-Hill book Company. 1997.

[112] Stollnitz, E., DeRose, T., Salesin, D. Wavelet for Computer Graphic. Morgan Kaufmann, San Fransisco. 1996.

[113] Strang G. Wavelet Transform versus Fourier Transforms. Bulletin of the Americam Mathematical Society. USA, v. 26, n. 2, p. 288-304, Abril 1993

[114] Daubechies I. Ten Lectures on Wavelets. Society for Industrial and Applied Mathematics, Philadelphia, 1992.

[115] Watkinson J. The MPEG Handbook. MPEG-1, MPEG-2, MPEG-4. Focal Press. Oxford. 2001. p.395.

[116] Wilson D. , Ghanbari M. Optimisation of two-layer SNR scalability for MPEG-2 video. Proc. Int. Conf. Acoust. Speech Signal Processing – ICASSP. p. 2637-2640, Munich, Alemanha, 1997.

[117] Mendes L., Costa M., Transmissão de vídeo utilizando o padrão MPEG-2 em duas camadas. XV Simpósio Brasileiro de Telecomunicações, pp. 585-589, Recife PE setembro 1997.

[118] Collins, Gerald W., Fundamentals of Digital Television Transmission. Wiley-Interscience. Illinois, USA. 2001. p.282.

[119] Whitaker, J. C., Standard Handbook of Video and Television Engineering. McGraw-Hill Professional. New York. 2000. p. 600.

[120] Whitaker, J. C. Video Display Engineering. McGraw-Hill Professional. New York. 2000. p. 459.

[121] Whitaker, J. C. Television Receivers : Digital Video for DTV, Cable and Satellite. McGraw-Hill Professional. New York. 2001. p. 586.

[122] Watkinson, J., Introduction to Digital Video. Focal Press. Oxford. 2001. p.320.

[123] Lajos Hanzo, et al Wireless Video Communications : Second to Third Generation and Beyond, John Wiley & Sons LTD. New York. 2001. p.1100.

[124] Prasad, L. Iyengar S. Wavelet Analysis with Applications to Image Processing. CRC Press LLC. USA. 1997. p. 279.

[125] Bovik, Al. Handbook of Image and Video Processing. Academic Press. San Diego. 2000. p 891.

[126] Petrou, M. ,Bosdogianni P. Image Processing : The Fundamentals. John Wiley & Sons LTD. New York.2000. p.333.

[127] Seul, M. , O´Gorman L. , Sammon M. Practical Algorithms for Image Analysis : Descriptions, Examples and Code. Cambridge University Press. Cambridge. 1999. p.295.

[128] Grimme, K. Digital Television Standardization and Strategies. Artech House. USA 2001. p.336.

[129] Watkinson, J. MPEG-2. Focal Press. Oxford. 2001. p.256.

[130] Nosratiania A. , Orchard M Multi-resolution backward video coding. IEEE International Conference on Image Processing, Washintong, USA, v.41, Outubro, 1995.

[131] Jesen A. , Cour-Harbo A. Ripples in Mathematics. Springer, New York USA, 2001. pp.246.

[132] Seul M. , O' Gorman L. , Sammon M. Practical Algorithms for Image Analysis. Cambridge University Press, USA, 1999. pp.296.

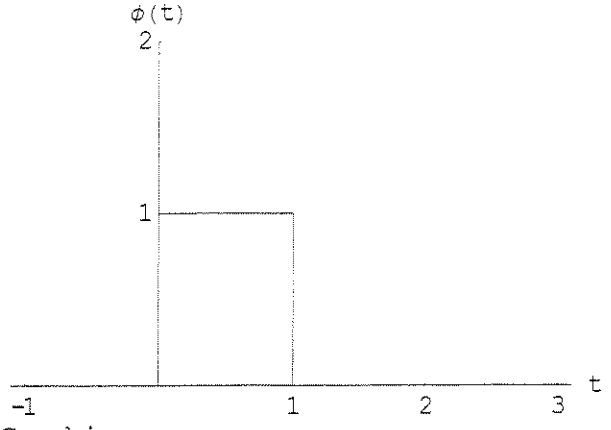
Anexo A. Experiências.

1. Análise de Multiresolução

Multiresolução de Haar

Função box, $\phi(t) = \begin{cases} 1 & \text{para } 0 \le t < 1 \\ 0 & \text{cc} \end{cases}$ para definir função escala

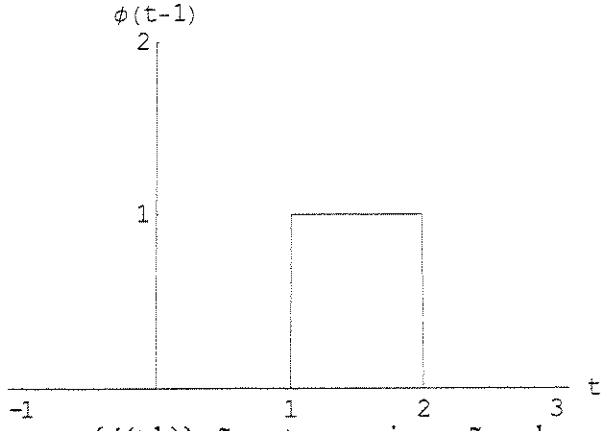
```
boxf[t_] := Which[0 <= t < 1, 1, True, 0]
Plot[boxf[t], {t, -1, 3}, PlotRange -> {0, 2},
Ticks -> {Automatic, {1, 2}},
AxesLabel -> {"t", "ϕ(t)"}]
```



- Graphics -

Trasladando ϕ(t)

```
Plot[boxf[t-1], {t, -1, 3}, PlotRange -> {0, 2},
Ticks -> {Automatic, {1, 2}},
AxesLabel -> {"t", "ϕ(t-1)"}];
```



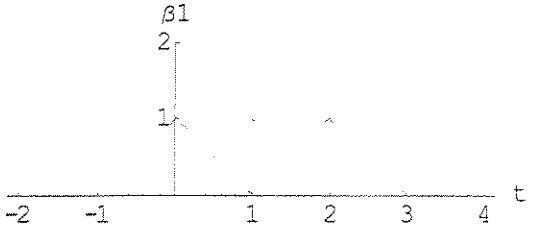
Observe, {ϕ(t-k)} são ortonormais ⇒ não solapamento entre ϕ(t-k) e ϕ(t-k') para k≠k'

Função linear *Picewise*

Função hat $\beta_1(t) = \begin{cases} 1-|t| & -1 \le t \le 1 \\ 0 & \text{cc} \end{cases}$

```
beta[t_] := Which[-1 < t < 1, 1 - Abs[t], True, 0]
```

```
Plot[{beta[t], beta[t - 1], beta[t - 2]}, {t, -2, 4},
PlotRange -> {0, 2}, Ticks -> {Automatic, {1, 2}},
AxesLabel -> {"t", " β1 "}, AspectRatio -> Automatic];
```



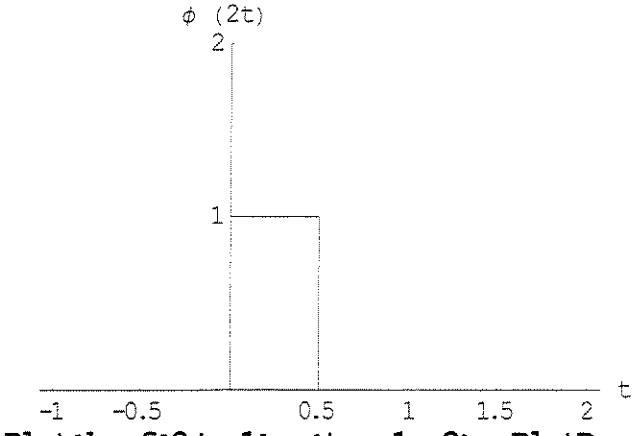
Observe, que $\{\beta_1(t-k)\}$ é não ortogonal $\Rightarrow \beta_1$ (t) é uma função escala não ortogonal

Multiresolução *Spline*

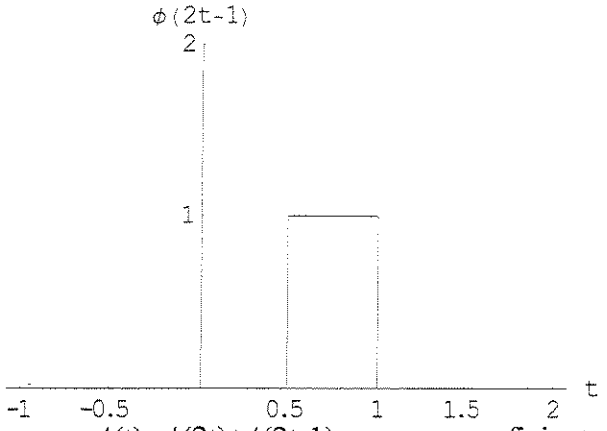
Os dois exemplos anteriores podem ser incluidos no análise de multiresolução Spline. Devido a,
Cada espaço V_0 é contituido pelas funções $f(t)$, que são polinômios de grau n para $[k, k+1)$ e tem n − 1 derivadas continuas
De modo que para, n = 0 ⇒ Multiresolução de Haar, n = 1 ⇒ Multiresolução das funções lineares picewise

ϕ e ψ

```
Plot[boxf[2 t], {t, -1, 2}, PlotRange -> {0, 2},
AxesLabel -> {"t", "ϕ (2t) "},
Ticks -> {Automatic, {1, 2}}];
```




```
Plot[boxf[2 t- 1], {t, -1, 2}, PlotRange -> {0, 2},
AxesLabel -> {"t", "ϕ(2t-1)"},
Ticks -> {Automatic, {1, 2}}];
```

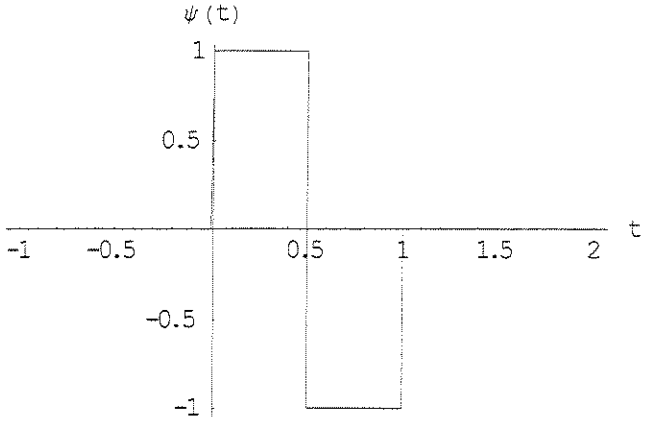


Observe, $\phi(t)=\phi(2t)+\phi(2t-1)$ e que os coeficientes do filtro são determinados a partir de (III.12). Calculando

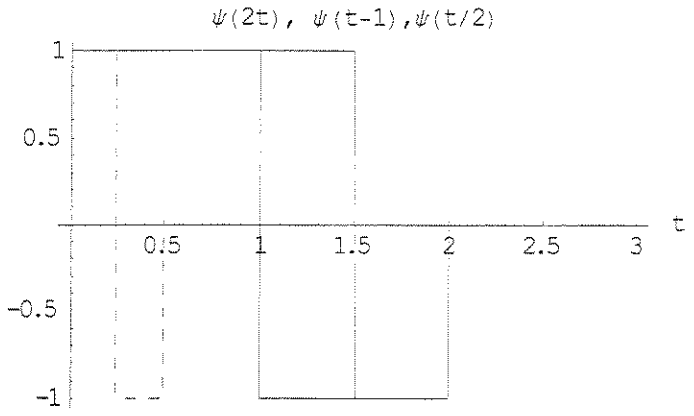
```
Sqrt[2.] NIntegrate[boxf[t] boxf[2 t], {t, 0, 1}]
0.707107
```

De (III.33) obtemos, $\psi(t) = \sum_k g_k \, \phi_{1\,k}(t) = \phi(2t) - \phi(2t-1)$, a wavelet de Haar,

```
haarPsi[t_] := boxf[2 t] - boxf[2 t - 1]
Plot[haarPsi[t], {t, -1, 2},
AxesLabel -> {"t", "ψ(t)"}]
```




```
- Graphics -
Plot[{haarPsi[2 t], haarPsi[t - 1], haarPsi[t / 2]},
{t, 0, 3}, PlotStyle ->
{Dashing[{0.02}], GrayLevel[0.5], {}},
PlotLabel -> "ψ(2t), ψ(t-1),ψ(t/2)",
AxesLabel -> {"t", ""}]
```

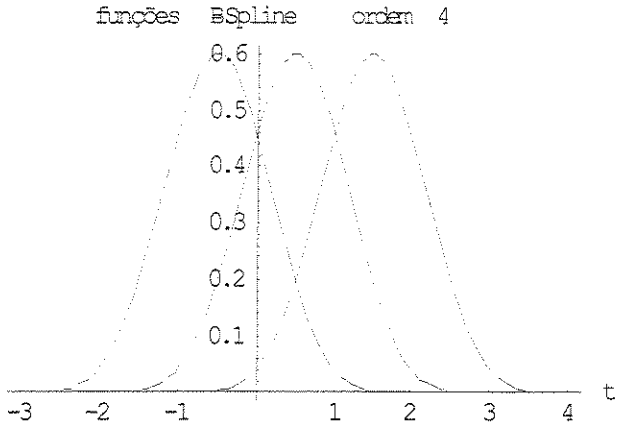


- Graphics -

2. Wavelets Ortogonais

Funções *Spline*

```
Needs["Wavelets`Wavelets`"]
Plot[Evaluate[{BSpline[4, t+1], BSpline[4, t],
BSpline[4, t-1]}], {t, -3, 4}, PlotRange -> All,
AxesLabel -> {"t", " funções B-Spline ordem 4 "}]
```




```
- Graphics -
```

Evaluando numericamente o grau de superposição,

```
NIntegrate[Evaluate[BSpline[4, t] BSpline[4, t-1]],
{t, -2, 4}]
0.243149
```

Ortogonalização das Funções B-Spline

Ver Mallat (1989) e Chui (1992)

Calculando sigma[n, w],

```
(sigma[2, w_] := 1/(4 Sin[w/2]^2);
sigma[n_, w_] := Module[{sig},
sig[x_] = (-1)^n D[1/Sin[x/2]^2, {x, n-2}]/(4 (n-1)!);
sig[w]])
```

Para n=1, calculamos b(ω),

```
(n = 1;
b = Expand[sigma[2 (n+1), w] (2 Sin[w/2])^(2 n+2)])
```

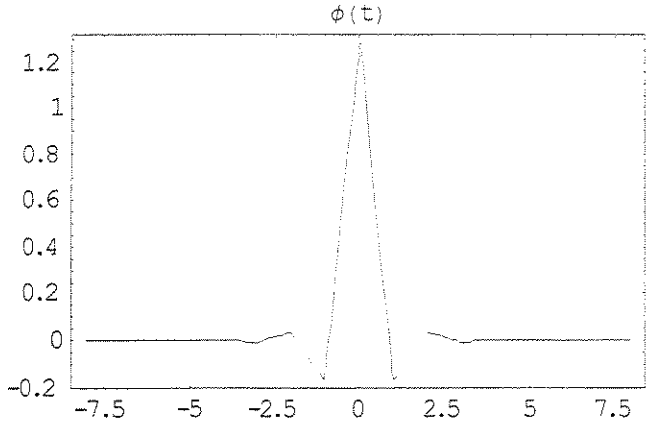
$$\frac{1}{3} + \frac{2}{3}\,\mathrm{Cos}\left[\frac{w}{2}\right]^2$$

Como $M(\omega)$, os $\{c_k\}$ são caculados atraves da serie coseno de Fourier,

```
Needs["Calculus`FourierTransform`"]
(kmax = 7;
clist = Table[NFourierCosSeriesCoefficient[
1/Sqrt[b], {w, 0, N[2 Pi]}, i], {i, 0, kmax}])
{1.29168, -0.349326, 0.0704202, -0.0157488, 0.00369589, -0.000891843, 0.000219154, -0.0000545
```

A função escala ϕ(t),

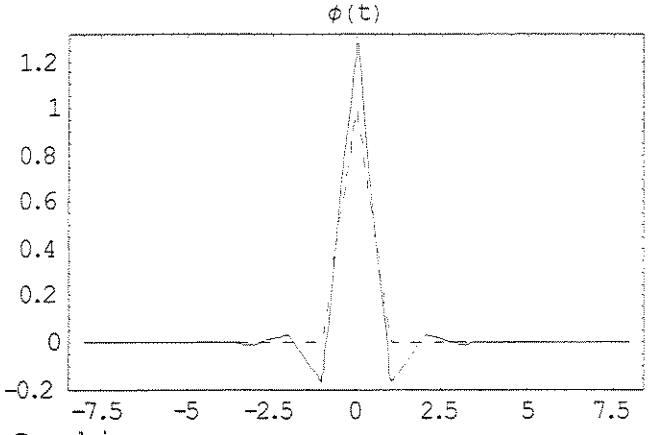
```
Plot[Evaluate[Table[BSpline[1, x - j], {j, -7, 7}].Join[
Reverse[Rest[clist]], {2 clist[[1]]}, Rest[clist]] / 2],
{x, -8, 8}, PlotRange -> All, Axes -> False,
Frame -> True, PlotLabel -> " φ(t) "]
```



- Graphics -

Comparando a função escala ortogonalizada (linha continua) com a original (não ortogonal),

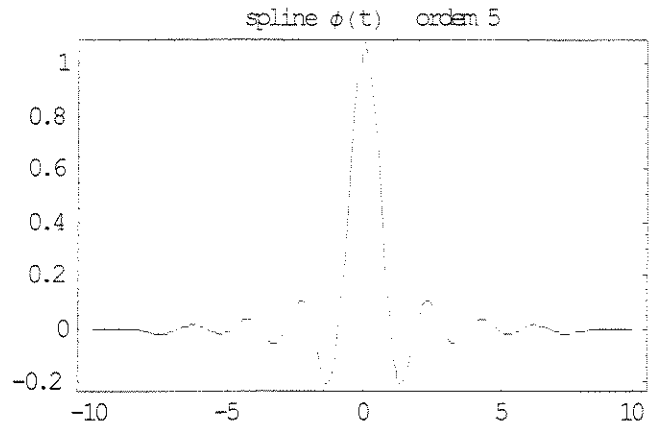
```
Show[%, Plot[BSpline[1, x], {x, -8, 8},
PlotStyle -> Dashing[{0.02}],
DisplayFunction -> Identity]]
```



- Graphics -

Função escala Spline ortogonal de ordem 5,

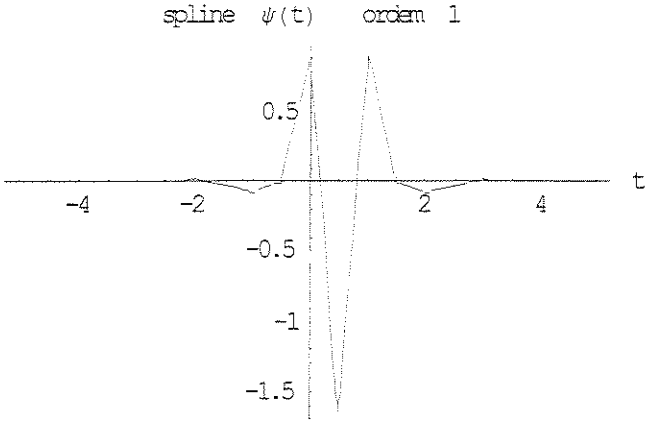
```
SplinePhi[5, t, 7];
Plot[Evaluate[%], {t, -10, 10}, PlotRange -> All,
Axes -> False, Frame -> True,
PlotLabel -> " spline  φ(t)  ordem  5 "]
```



- Graphics -

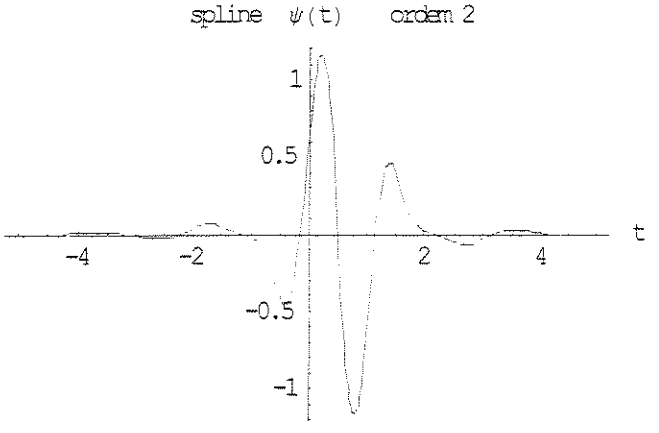
Wavelets,

```
Plot[Evaluate[SplinePsi[1, t, 5]], {t, -5, 5},
AxesLabel -> {"t", "spline    ψ(t)   ordem     1"},
PlotRange -> All]
```



- Graphics -

```
Plot[Evaluate[SplinePsi[2, t, 8]], {t, -5, 5},
AxesLabel -> {"t", "   spline    ψ(t)   ordem    2"},
PlotRange -> All]
```



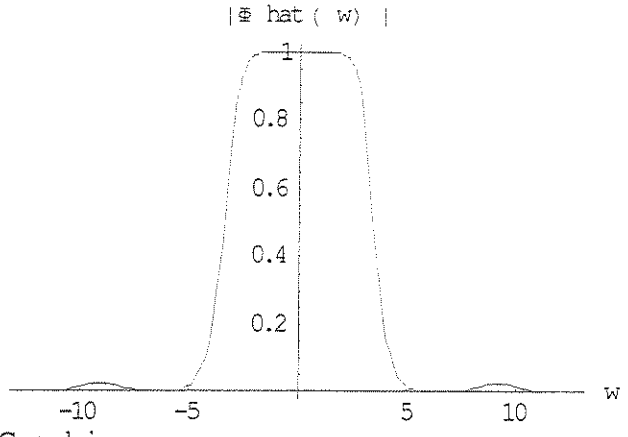
- Graphics -

ϕ e ψ no espaço de Fourier

A partir de (III.44),

função escala $\Phi(\omega)$, para n=2

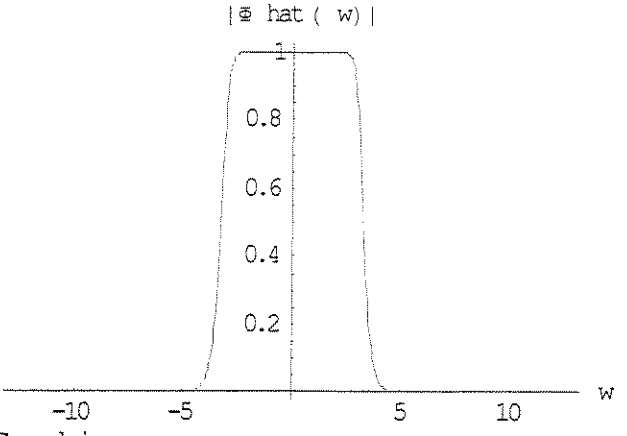
```
(splinePhiHat[n_, 0] = 1;
splinePhiHat[n_, w_] := 1 / Sqrt[sigma[2 n + 2, w] w^(2 n + 2)])
Plot[Evaluate[splinePhiHat[2, w]], {w, -4 Pi, 4 Pi},
PlotRange -> All, AxesLabel -> {"w", "| Φ  hat  ( w ) |"}]
```



- Graphics -

função escala $\Phi(\omega)$, para n=5

```
Plot[Evaluate[splinePhiHat[5, w]], {w, -4 Pi, 4 Pi},
PlotRange -> All, AxesLabel -> {"w", "| Φ  hat  ( w )|"}]
```



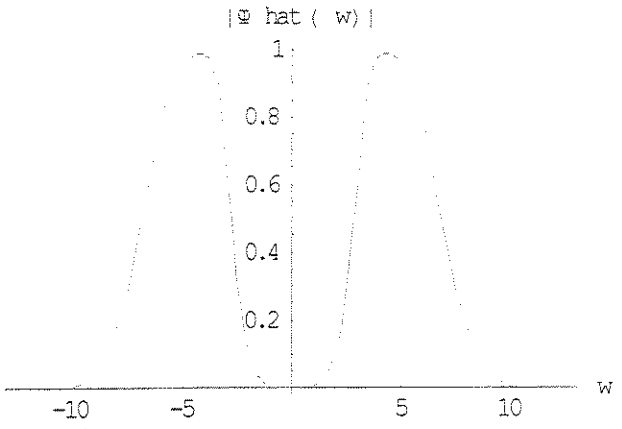
- Graphics -

wavelet,

```
splinePsiHat[n_, w_] := Module[{sig},
sig[x_] = sigma[2 n + 2, x];
Chop[Sqrt[Simplify[sig[w / 2 + Pi] / sig[w]] / sig[w / 2] / w^(2 n + 2)]]]
```

wavelet $\Psi(\omega)$de ordem 2,

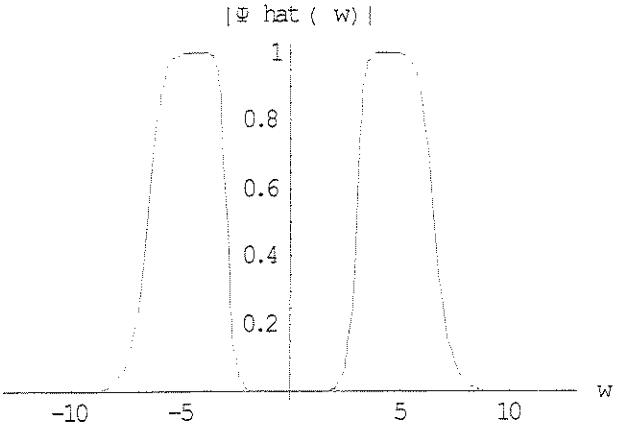
```
Plot[Evaluate[splinePsiHat[2, w]], {w, -4 Pi, 4 Pi},
PlotRange -> All, AxesLabel -> {"w", "| Ψ  hat  ( w )|"}]
```



- Graphics -

wavelet Ψ (ω) de ordem 5,

```
Plot[Evaluate[splinePsiHat[5, w]], {w, -4 Pi, 4 Pi},
PlotRange -> All, AxesLabel -> {"w", "| Ψ  hat  ( w )|"}]
```



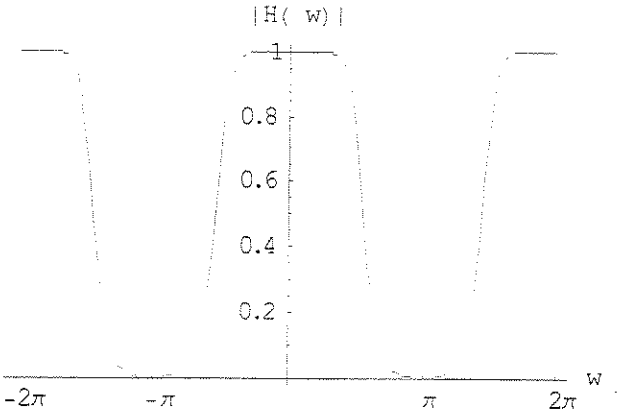
- Graphics -

De (III.21) podemos obter a magnitude de |H(ω)|,

```
splineResponseH[n_, w_] := Module[{sig},
sig[x_] = sigma[2 n+ 2, x];
Sqrt[Simplify[sig[w] / sig[2 w]]] / 2^(n + 1)]
```

para n=2

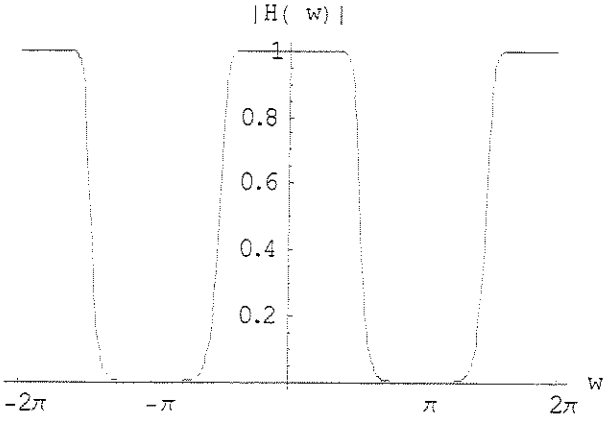
```
Plot[Evaluate[splineResponseH[2, w]], {w, -2 Pi, 2 Pi},
Ticks -> {Range[-2 Pi, 2 Pi, Pi], Automatic},
AxesLabel -> {"w", "| H ( w )|"}]
```



- Graphics -

para n=5

```
Plot[Evaluate[splineResponseH[5, w]], {w, -2 Pi, 2 Pi},
Ticks -> {Range[-2 Pi, 2 Pi, Pi], Automatic},
AxesLabel -> {"w", "| H ( w )|"}]
```



- Graphics -

Observe que a $|H(\omega)|$ é de tipo passa baixas. A reposta de $|G(\omega)|$, calcula – se $G(\omega) = e^{-i\omega} H^*(\omega + \pi)$ e é de tipo pass

3. Wavelet de Daubechies

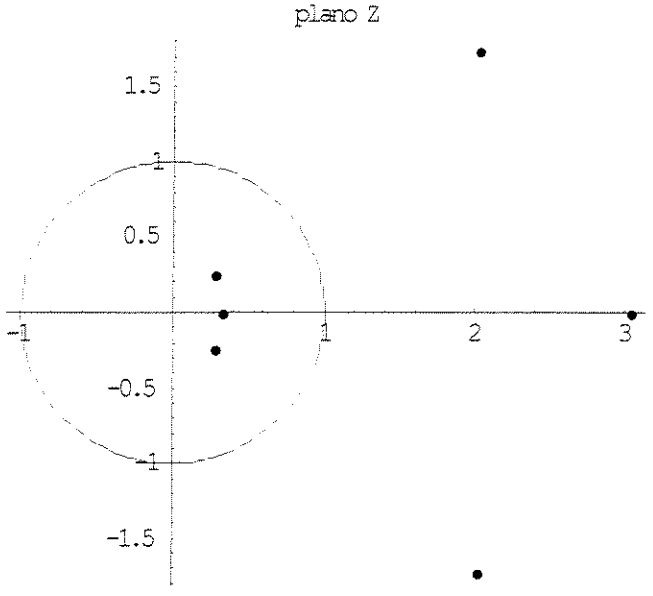
Filtros de Daubechies

Ver Seção III.9.1,

```
Needs["Wavelets`Wavelets`"]
p[n_, z_] := Sum[Binomial[n+k-1, k] *
((1-(z+1/z)/2)/2)^k, {k, 0, n-1}]
(n = 4;
roots = Last /@ List @@
NRoots[z^(n-1) p[n, z] == 0, z])
```

{0.284096 - 0.243228 I, 0.284096 + 0.243228 I, 0.328876, 2.03114 - 1.73895 I, 2.03114 + 1.73895 I, 3.04066}

```
Show[Graphics[Circle[{0, 0}, 1]],
Graphics[{PointSize[0.015],
Point[{Re[#], Im[#]}] & /@ roots}],
AspectRatio -> Automatic, Axes -> True,
PlotLabel -> " plano - Z "]
```



- Graphics -

São selecionadas as raízes que estão dentro do círculo unitário,

```
roots = Select[roots, Abs[#] < 1 &]
```

{0.284096 - 0.243228 I, 0.284096 + 0.243228 I, 0.328876}

$(z - z_j)(z - \overline{z}_j)(z - r_j)$

```
Apply[Times, z - roots]
```

(-0.328876 + z) ((-0.284096 - 0.243228 I) + z) ((-0.284096 + 0.243228 I) + z)

coeficientes desejados do filtro ,

```
(coef = Re[CoefficientList[(1+z)^n %, z]];
Reverse[coef] / (Apply[Plus, coef] / N[Sqrt[2]]))
```

{0.230378, 0.714847, 0.630881, -0.0279838, -0.187035, 0.0308414, 0.032883, -0.0105974}

Para N=3

```
h3 = DaubechiesFilter[3]
{{0.332671, 0.806892, 0.459878, -0.135011, -0.0854413, 0.0352263}, 0}
```

para N=11, com presição 28,

```
DaubechiesFilter[11, 28]
```

```
{{0.01869429776147108403, 0.1440670211506245128, 0.44989764356045335, 0.685686774916200511,
  0.41196436894790746 3, -0.162275245027490362 2, -0.27423084681794696 1, 0.06604358819668319 2, 0.14981201246637
  -0.04647995511668418 7, -0.06643878569502520 5, 0.03133509021904607 6, 0.02084090436018106 30, -0.0153648209062
  -0.00334085887301444 56, 0.00492841765605904 11, -0.00030859285881514317, -0.00089302325066626461,
  0.00024915252355282 3499, 0.00005443907469936 84717, -0.00003463498418698 49955, 4.4942742772365101 0×10^-6}, 0}
```

ϕ e ψ

Para N=2,

Os coeficientes do filtro são calculados,

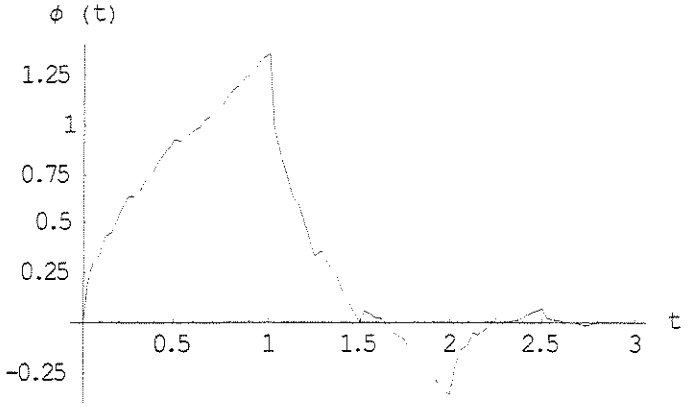
```
d2 = DaubechiesFilter[2]
{{0.482963, 0.836516, 0.224144, -0.12941}, 0}
```

função escala com deslocamentos diádicos $k/2^5$,

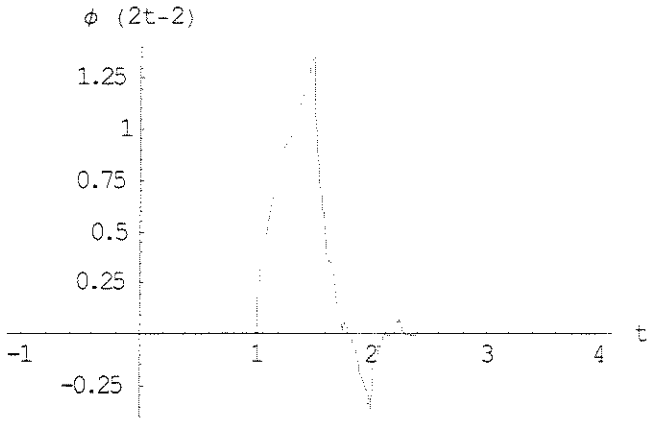
```
philist = ScalingFunction[d2, 5];
Short[philist]
```

$\left\{\{0, 0.\}, \left\{\frac{1}{32}, 0.20305\right\}, \text{<<93>>}, \left\{\frac{95}{32}, 0.000075148\right\}, \{3, 0\}\right\}$

```
ListPlot[philist, PlotJoined -> True,
AxesLabel -> {"t", " ϕ (t) "}]
```

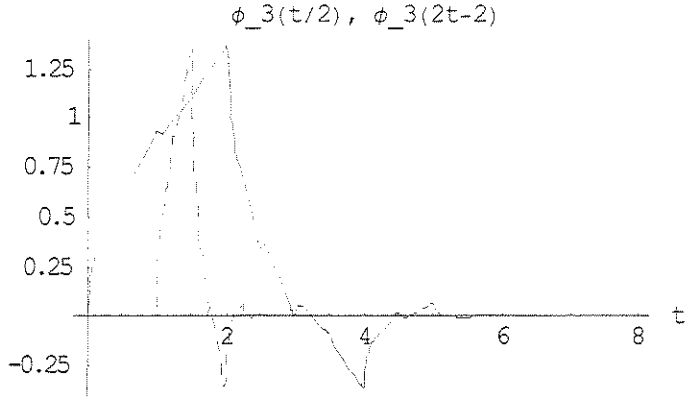



```
- Graphics -
phi = Interpolation[Join[{{-20, 0}}, philist,
{{20, 0}}], InterpolationOrder -> 1]
InterpolatingFunction[{{-20., 20.}}, <>]
Plot[phi[2 t - 2], {t, -1, 4}, PlotRange -> All,
AxesLabel -> {"t", " ϕ (2t-2) "}]
```



- Graphics -

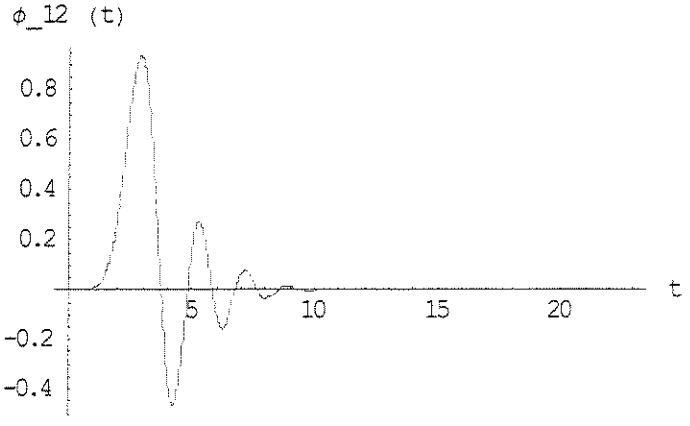
```
Plot[{phi[t/2], phi[2t-2]}, {t, 0, 8},
PlotStyle -> {{}, Dashing[{0.02}]}, PlotRange -> All,
AxesLabel -> {"t", ""},
PlotLabel -> "ϕ_ 3 (t/2), ϕ_ 3 (2t-2)"]
```



- Graphics -

Para N=12,

```
d12 = DaubechiesFilter[12];
ListPlot[ScalingFunction[d12, 6], PlotRange -> All,
PlotJoined -> True, AxesLabel -> {"t", "ϕ_ 12 (t)"}]
```



- Graphics -

wavelet com deslocamentos diádicos $k/2^6$,

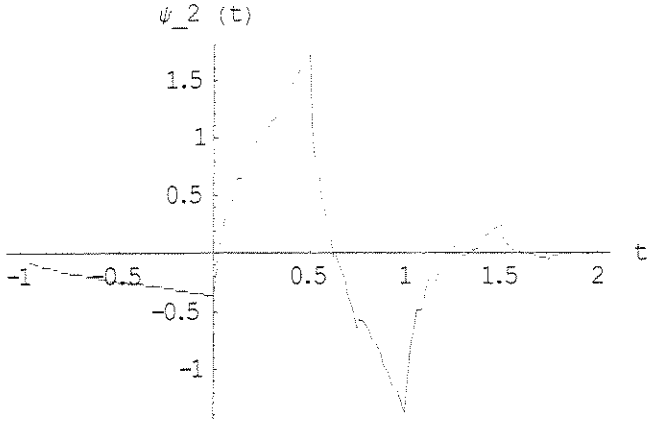
```
psi2 = Wavelet[d2, 6];
```

Para N=2,

```
Short[psi2]
```

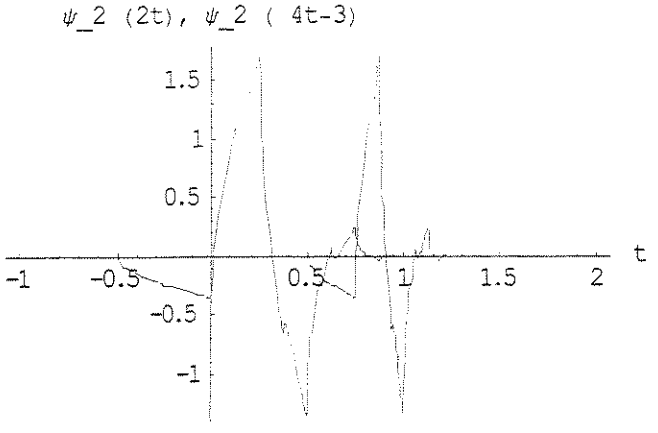
$$\left\{\{-1,\ 0.\},\ \left\{-\frac{63}{64},\ -0.0371607\right\},\ \text{<<189>>},\ \left\{\frac{127}{64},\ -0.000051327\right\},\ \{2,\ 0\}\right\}$$

```
ListPlot[psi2, PlotJoined -> True,
PlotRange -> All, AxesLabel -> {"t", "ψ_ 2 (t) "}]
```



- Graphics -

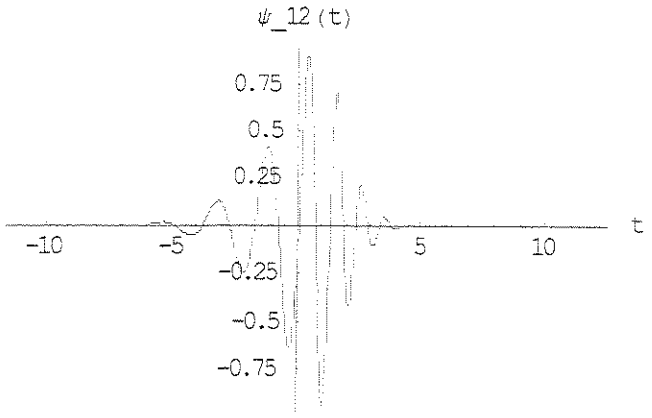
```
psi2 = Wavelet[d2, 6, Interpolation -> True]
InterpolatingFunction[{{-100., 100.}}, <>]
Plot[{psi2[2 t], psi2[4 t - 3]}, {t, -1, 2},
PlotRange -> All,
AxesLabel -> {"t", "ψ_ 2 (2t), ψ_ 2 ( 4 t-3 )"}]
```



- Graphics -

Para N=12,

```
ListPlot[Wavelet[d12, 6], PlotJoined -> True,
PlotRange -> All, AxesLabel -> {"t", " ψ_ 12(t)"}]
```



- Graphics -

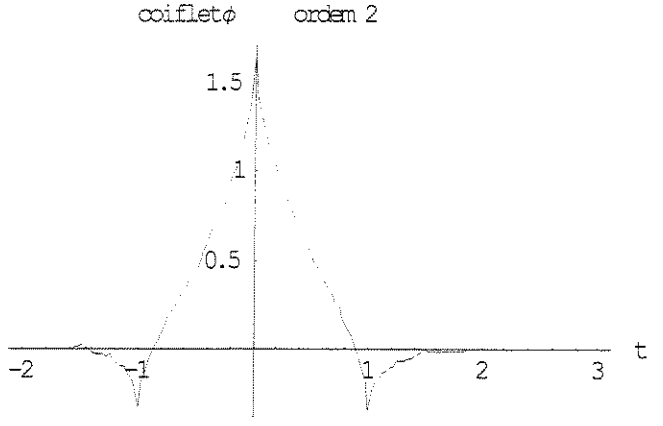
Coiflets

para N=2,

```
c2 = CoifletFilter[2, 18]
{{-0.072732619512526448, 0.337897662457481770,
  0.85257202021160042, 0.384864846864857747, -0.072732619512526448, -0.015655728135791925}, -2}
```

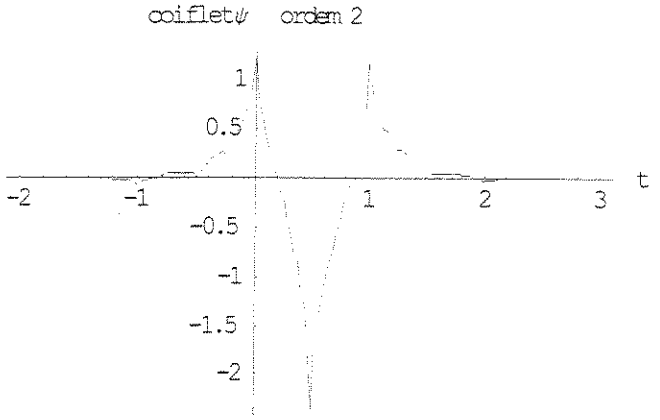
função escala,

```
ListPlot[ScalingFunction[c2, 5],
PlotJoined -> True, PlotRange -> All,
AxesLabel -> {"t", "coiflet    ϕ    ordem    2"}];
```



wavelet,

```
ListPlot[Wavelet[c2, 5],
PlotJoined -> True, PlotRange -> All,
AxesLabel -> {"t", "coiflet    ψ  ordem    2"}];
```

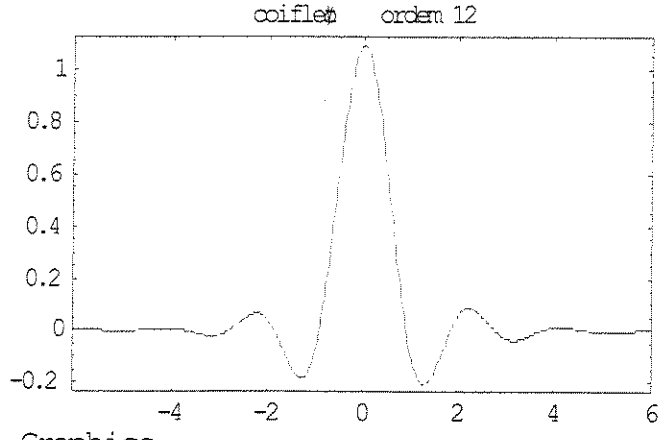


para N=12

```
c12 = CoifletFilter[12, 20];
```

função escala,

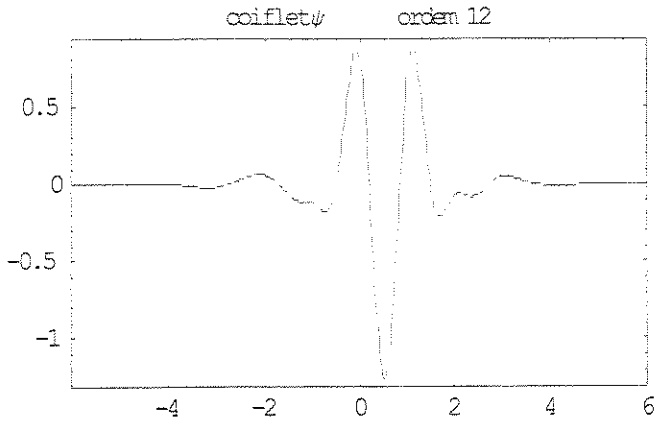
```
ListPlot[ScalingFunction[c12, 5], Axes -> False,
Frame -> True, PlotRange -> {{-6, 6}, All},
PlotJoined -> True,
PlotLabel -> "coiflet  φ  ordem  12"]
```



- Graphics -

wavelet,

```
ListPlot[Wavelet[c12, 5], Axes -> False,
Frame -> True, PlotRange -> {{-6, 6}, All},
PlotJoined -> True,
PlotLabel -> "coiflet  ψ  ordem  12"]
```



- Graphics -

4. Transformada Wavelet 2-D

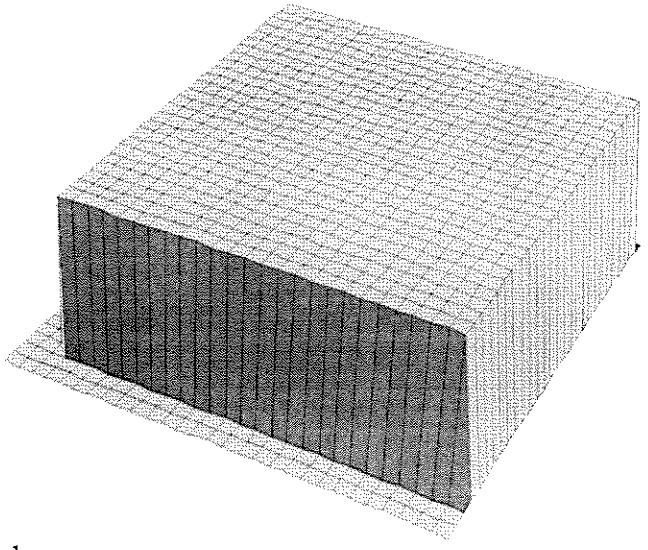
```
Needs["Wavelets`Wavelets`"]
```

Primeiro geramos função escalar e a wavelet -mãe em 1-D,

```
(h1 = DaubechiesFilter[1];
 phi = ScalingFunction[h1, 6, Interpolation -> True];
 psi = Wavelet[h1, 6, Interpolation -> True];)
```

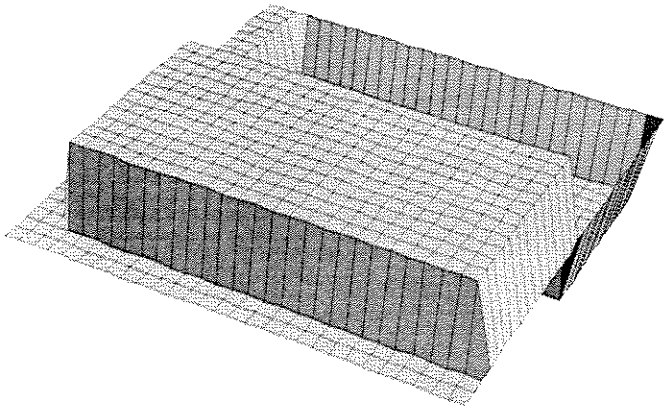
função escala de Haar,

```
Plot3D[Evaluate[phi[x] phi[y]], {x, -0.1, 1},
{y, -0.1, 1}, PlotPoints -> 30, PlotRange -> All,
Boxed -> False, Axes -> False];
```



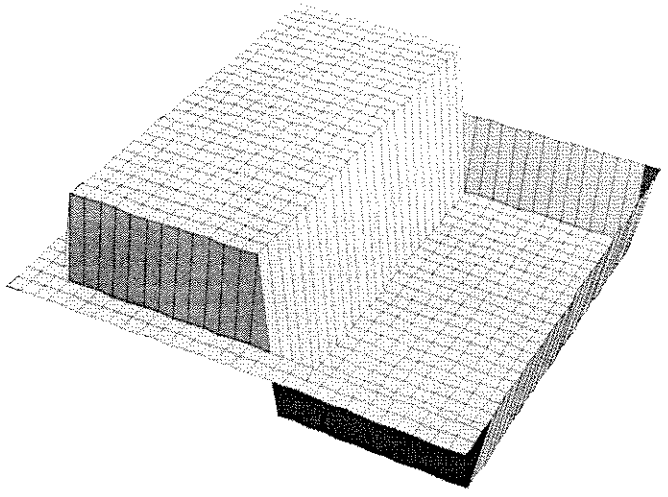
$\psi^1(x, y)$,

```
Plot3D[Evaluate[phi[x] psi[y]], {x, -0.1, 1},
{y, -0.1, 1}, PlotPoints -> 30, PlotRange -> All,
Boxed -> False, Axes -> False];
```



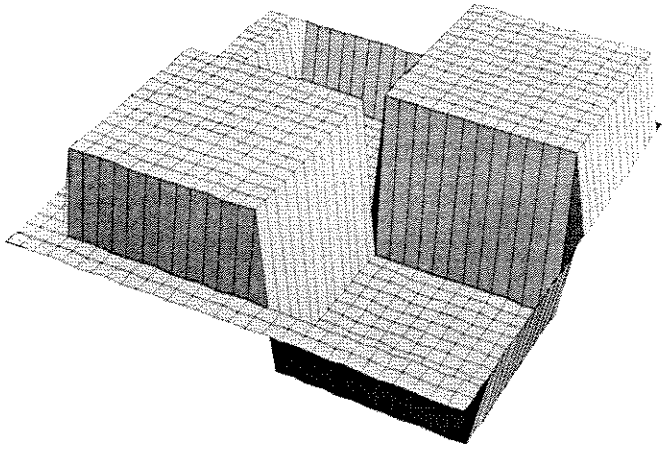
$\psi^2(x, y)$ que é $\psi^1(y, x)$,

```
Plot3D[Evaluate[psi[x] phi[y]], {x, -0.1, 1},
{y, -0.1, 1}, PlotPoints -> 30, PlotRange -> All,
Boxed -> False, Axes -> False];
```



$\psi^3(x, y)$,

```
Plot3D[Evaluate[psi[x] psi[y]], {x, -0.1, 1},
{y, -0.1, 1}, PlotPoints -> 30, PlotRange -> All,
Boxed -> False, Axes -> False];
```

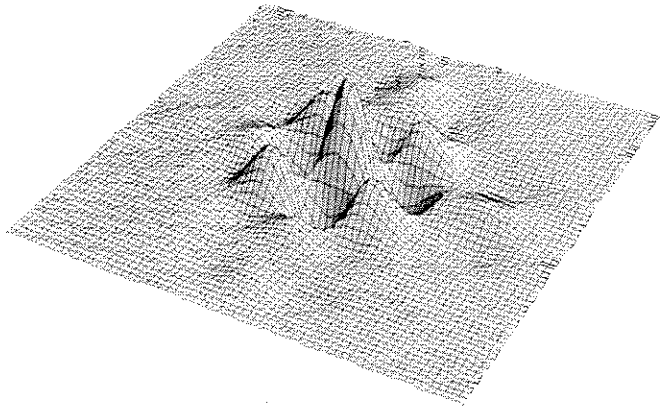


para N=8, wavelet *least asymmetric*
ψ em 1-D,

```
(s8 = LeastAsymmetricFilter[8];
psi = Wavelet[s8, 7, Interpolation -> True];)
```

$\psi^3(x, y)$,

```
Plot3D[Evaluate[psi[x] psi[y]], {x, -2, 3}, {y, -2, 3},
PlotPoints -> 70, PlotRange -> All,
Boxed -> False, Axes -> False];
```



5. Wavelets Biortogonais

Wavelet *Spline* Biortogonais

```
Needs["Wavelets`Wavelets`"]
```

Para N=3 e $\tilde{N}$=1 ,

```
b31 = BiorthogonalSplineFilter[3, 1]
```

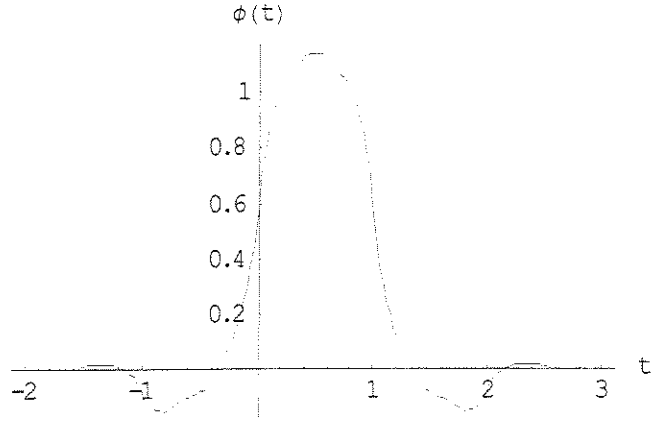
$$\left\{\left\{\left\{-\frac{1}{8\sqrt{2}}, \frac{1}{8\sqrt{2}}, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, \frac{1}{8\sqrt{2}}, -\frac{1}{8\sqrt{2}}\right\}, -2\right\}, \left\{\left\{\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right\}, 0\right\}\right\}$$

```
b31 = MapAt[N, #, 1] & /@ b31
```

```
{{{-0.0883883, 0.0883883, 0.707107, 0.707107, 0.0883883, -0.0883883}, -2}, {{0.707107, 0.707107},
```

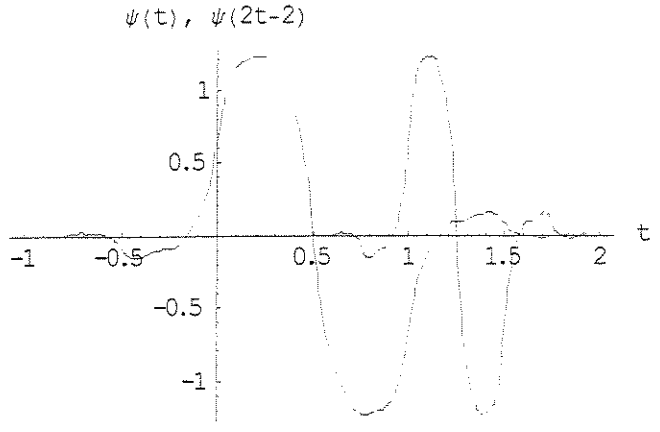
função escala ϕ,ψ

```
phi = ScalingFunction[b31, 5];
ListPlot[phi, AxesLabel -> {"t", "ϕ(t)"},
PlotJoined -> True, PlotRange -> All];
```



wavelet ψ,

```
psi = Wavelet[b31, 5, Interpolation -> True];
Plot[{psi[t], psi[2 t - 2]}, {t, -1, 2}, PlotRange -> All,
AxesLabel -> {"t", "ψ(t), ψ(2t-2)"}]
```

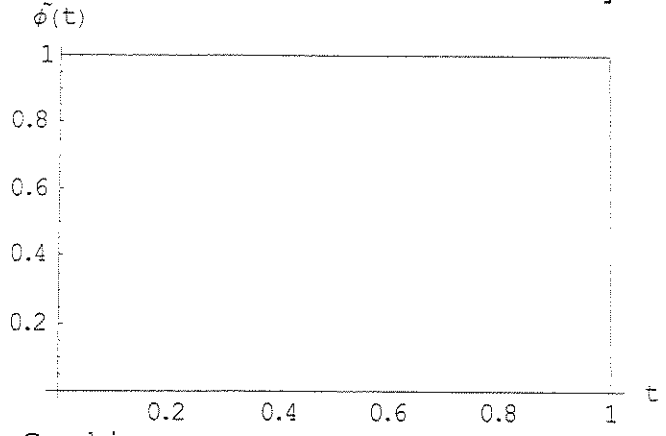



```
- Graphics -
```

função escala dual $\tilde{\phi}$,

```
phitilde = ScalingFunction[Reverse[b31], 7];
```

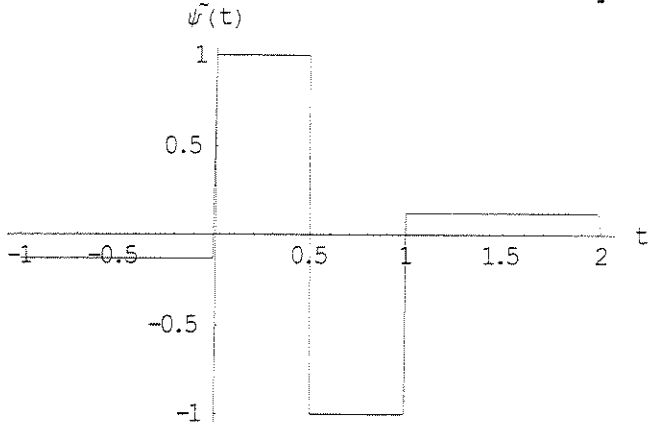
```
ListPlot[phitilde, AxesLabel -> {"t", "φ̃(t)"},
PlotJoined -> True, PlotRange -> All]
```



- Graphics -

wavelet dual $\tilde{\psi}$,

```
ListPlot[Wavelet[Reverse[b31], 7],
AxesLabel -> {"t", "ψ̃(t)"},
PlotJoined -> True, PlotRange -> All]
```



- Graphics -

Transformada Wavelet Biortogonal

filtro Spline biortogonal da ordem {10,4},

```
b = BiorthogonalSplineFilter[10, 4];
```

Gerando um conjunto aleatório de dados,

```
(SeedRandom[38437];
data = Table[Random[], {40}];)
```

aplicando a TW biortogonal,

```
wtdata = WaveletTransform[data, N[b]];
```

Observe que a TW não ortogonal não preserva a "norma" dos dados, porém os filtros biortogonais permitem reconstrução perfeita

```
Flatten[wtdata].Flatten[wtdata] - data.data
```

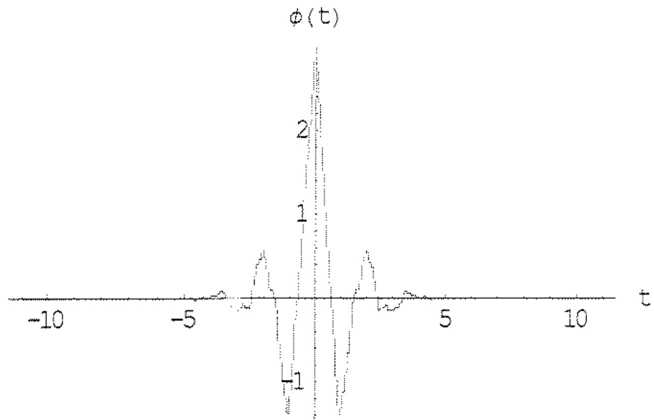
```
4.61903
```

```
Chop[InverseWaveletTransform[wtdata, N[b]] - data]
```

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, C

$\phi(t)$,

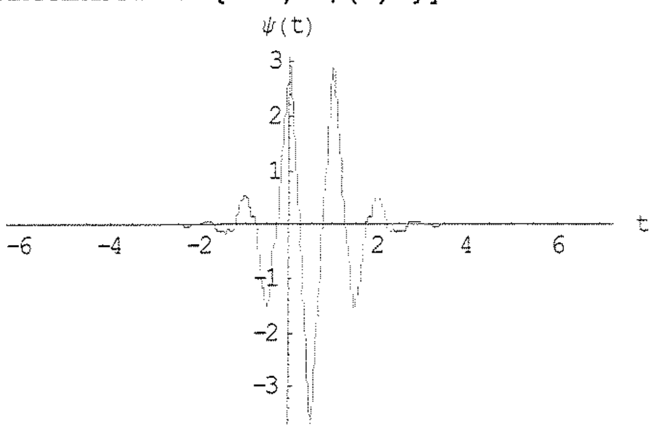
```
ListPlot[ScalingFunction[N[b], 7,
InverseTransform -> True],
PlotJoined -> True, PlotRange -> All,
AxesLabel -> {"t", "φ(t)"}]
```



- Graphics -

$\psi(t)$,

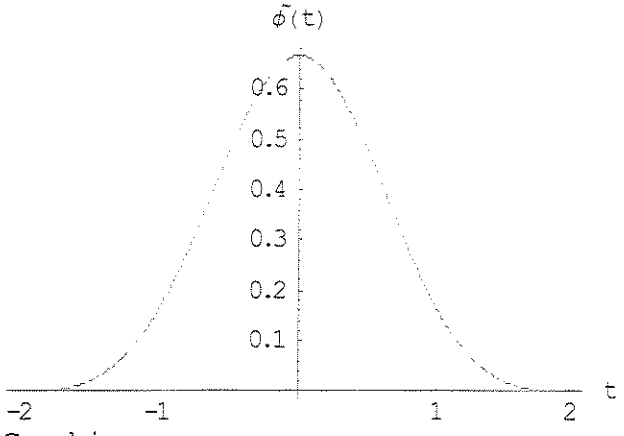
```
ListPlot[Wavelet[N[b], 7, InverseTransform -> True],
PlotJoined -> True, PlotRange -> All,
AxesLabel -> {"t", "ψ(t)"}]
```



- Graphics -

$\tilde{\phi}(t)$,

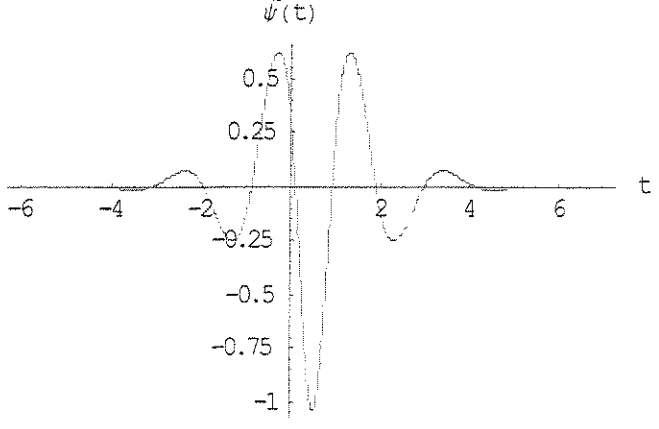
```
ListPlot[ScalingFunction[Reverse[N[b]], 7, InverseTransform -> True],
PlotJoined -> True, PlotRange -> All,
AxesLabel -> {"t", "φ̃(t)"}]
```



- Graphics -

$\tilde{\psi}(t)$,

```
ListPlot[Wavelet[Reverse[N[b]], 7,
InverseTransform -> True],
PlotJoined -> True, PlotRange -> All,
AxesLabel -> {"t", "ψ̃(t)"}]
```



- Graphics -

6. A TW NO PROCESSAMENTO DE IMAGENS.

Compressão de Imagens

O exemplo seguinte demonstra o uso da decomposição wavelet na compressão de imagens . O arquivo ***swphoto. eps*** contém uma *scanner* de uma imagem

```
Needs["Wavelets`Wavelets`"]
```

Lê arquivo e converte numa matriz,

```
data = Reverse[Partition[ToExpression /@
(StringJoin[Join[{"16^^"}, #]] & /@
Partition[Characters[StringJoin @@ Drop[Drop[
ReadList["Wavelets/Data/swphoto.eps", String],
22], -3]], 2]), 250]];
Dimensions[data]
```

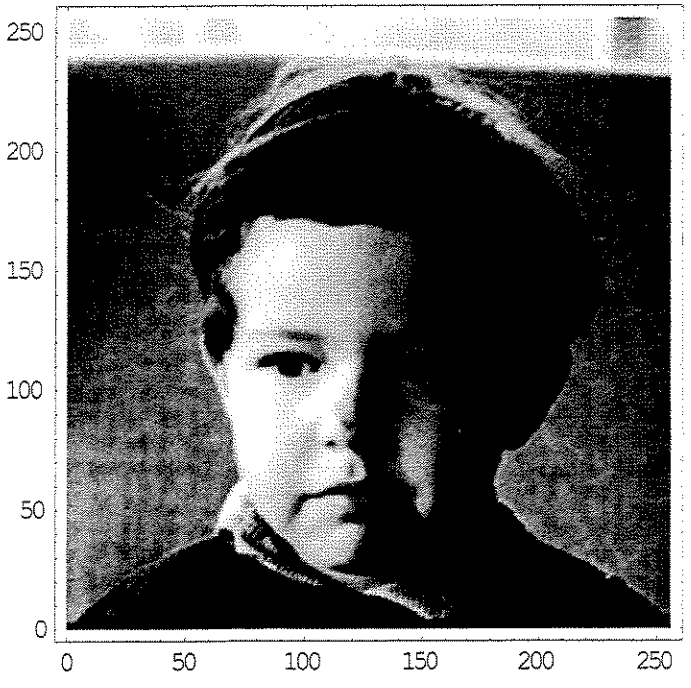
```
{277, 250}
```

Nós tiramos as filas extras na parte superior e colocamos 3 colunas extras ambas na esquerda e à direita de forma que o conjunto de dados seja {256,256}.

```
(data = Transpose[Join[Table[#[[1]], {3}], #,
Table[#[[-1]], {3}]] &[Transpose[Drop[data, 21]]]];
Dimensions[data])
```

```
{256, 256}
```

```
ListDensityPlot[data, Mesh -> False]
```



- DensityGraphics -

Usando um filtro *least asymmetric* de ordem 4. Mudamos o índice começando de 0 a -3

```
(s4 = LeastAsymmetricFilter[4];
s4 = MapAt[-3 &, s4, {2}])
```

{{-0.0757657, -0.0296355, 0.497619, 0.803739, 0.297858, -0.0992195, -0.012604, 0.032223

Aplicando TW

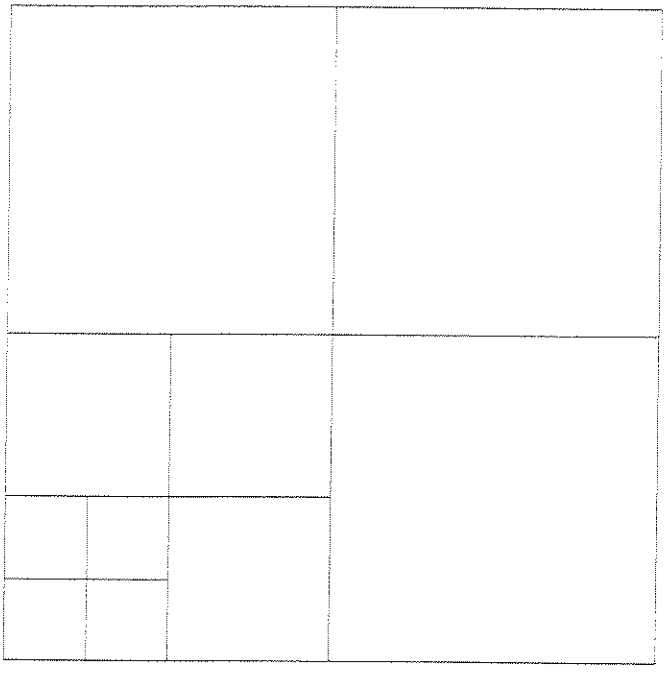
```
wtdata = WaveletTransform[data, s4, 3,
BoundaryCondition -> Fixed];
```

A saída da TW é organizada de acordo com os níveis de resolução. Note que a cada nível de resolução há três matrizes de detalhe.

```
Dimensions /@ wtdata
```

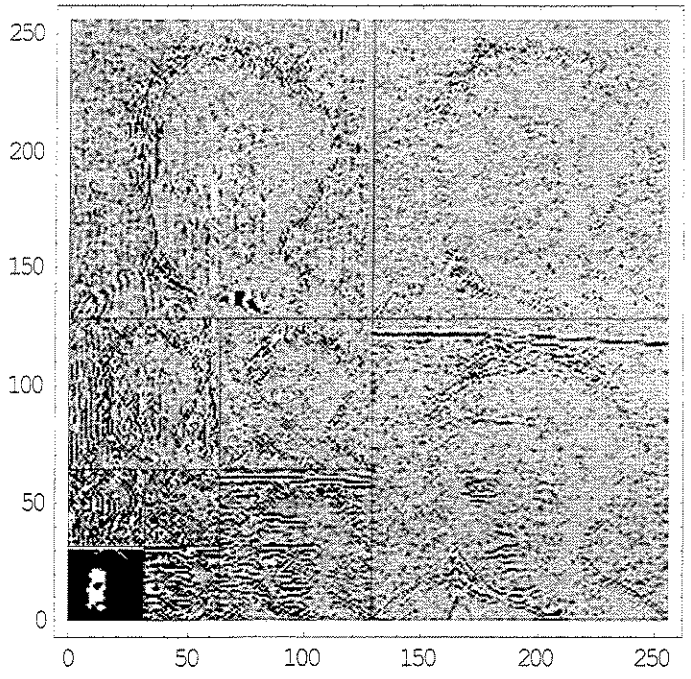
{{32, 32}, {3, 32, 32}, {3, 64, 64}, {3, 128, 128}}

```
ShowBasisPosition2D[wtdata, AspectRatio -> Automatic]
```



- Graphics -

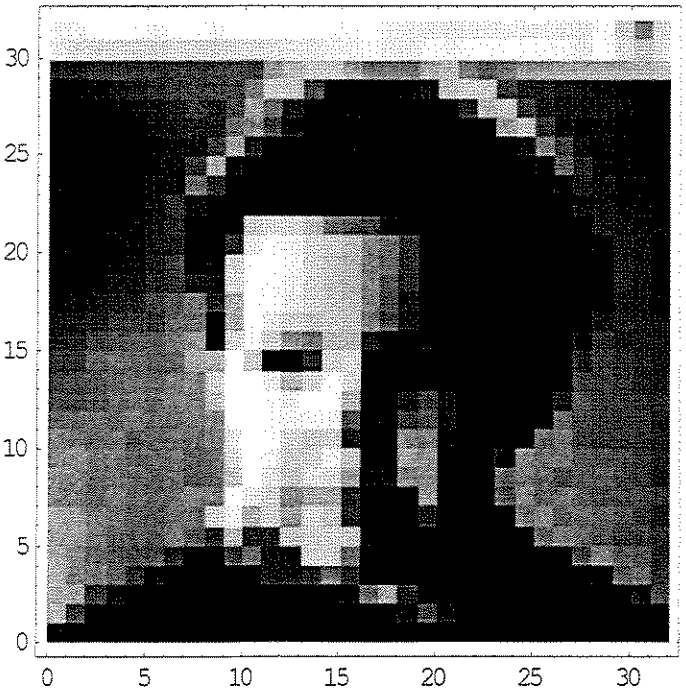
```
PlotCoefficients2D[wtdata]
```



- Graphics -

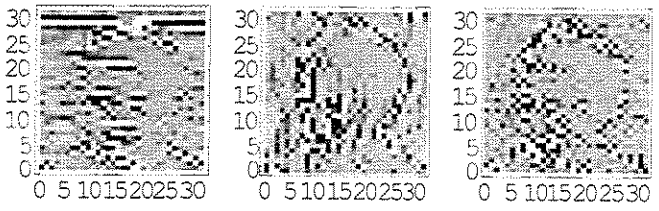
Colocando cada matriz separadamente.

```
ListDensityPlot[wtdata[[1]], Mesh -> False]
```



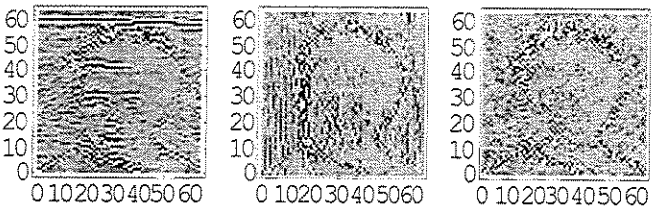
- DensityGraphics -

```
Show[GraphicsArray[ListDensityPlot[
#, Mesh -> False, DisplayFunction -> Identity] & /@
wtdata[[2]]]]
```



- GraphicsArray -

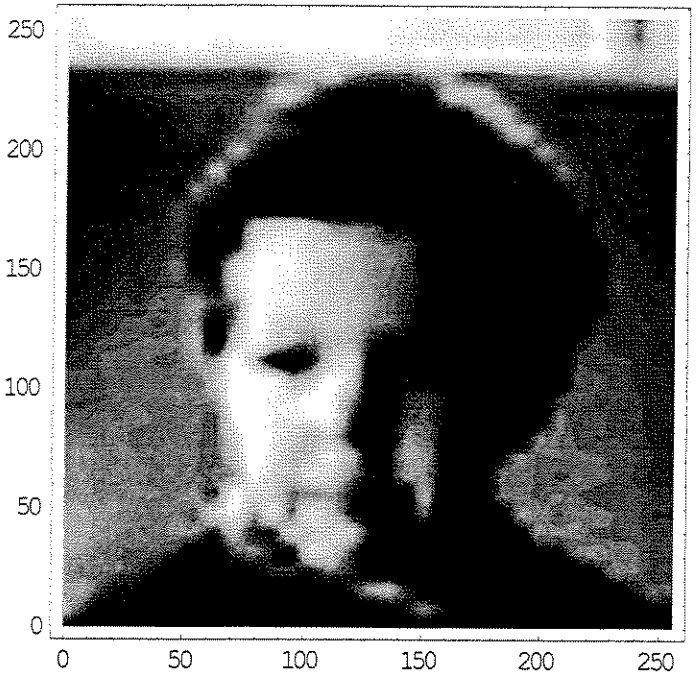
```
Show[GraphicsArray[ListDensityPlot[
#, Mesh -> False, DisplayFunction -> Identity] & /@
wtdata[[3]]]]
```



- GraphicsArray -

Transformação Inversa,

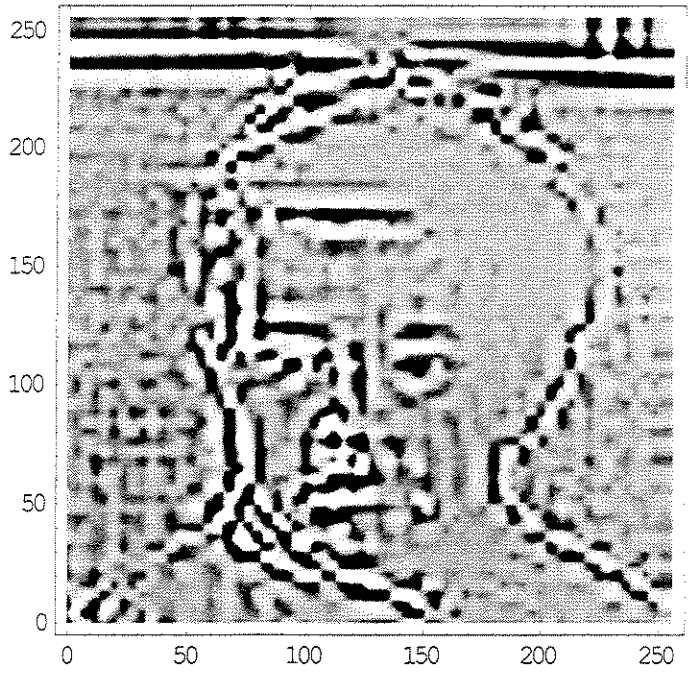
```
temp1 = InverseWaveletTransform[
wtdata - MapAt[# - # &, wtdata, {1}], s4,
BoundaryCondition -> Fixed];
ListDensityPlot[temp1, Mesh -> False]
```



- DensityGraphics -

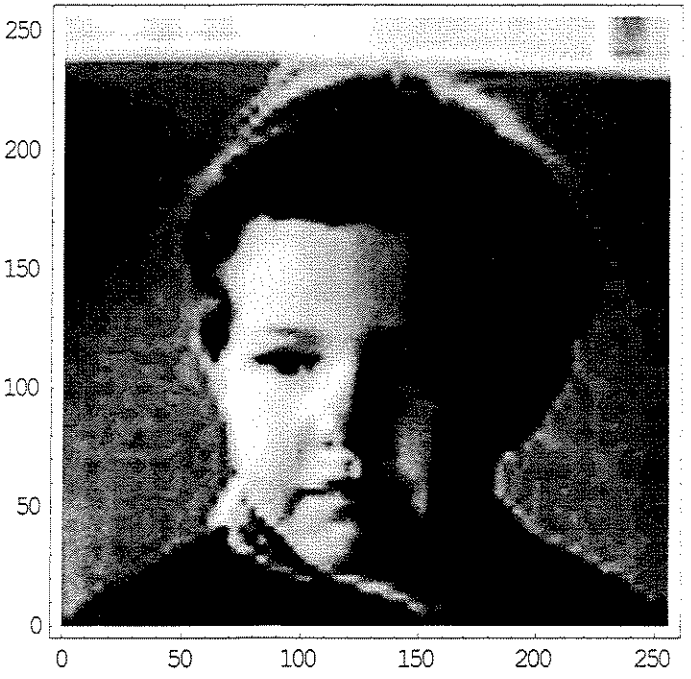
```
InverseWaveletTransform[
wtdata - MapAt[# - # &, wtdata, {2}], s4,
BoundaryCondition -> Fixed];
```

```
ListDensityPlot[%, Mesh -> False]
```



- DensityGraphics -

```
ListDensityPlot[temp1 + %%, Mesh -> False]
```



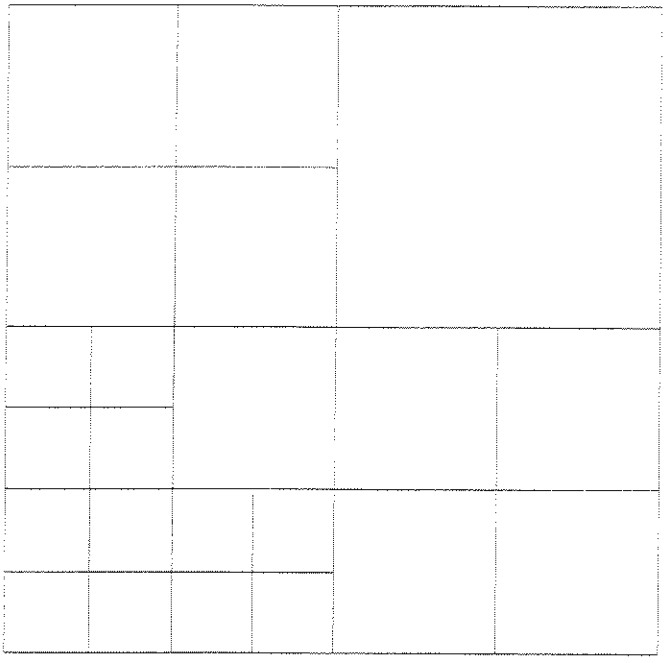
- DensityGraphics -

Aplicando Wavelet Packet

```
wpdata = WaveletPacketTransform[data, s4, 3,
BoundaryCondition -> Fixed];
```

Comparando, a Wavelet Packet decompõe algum subspaces mais .

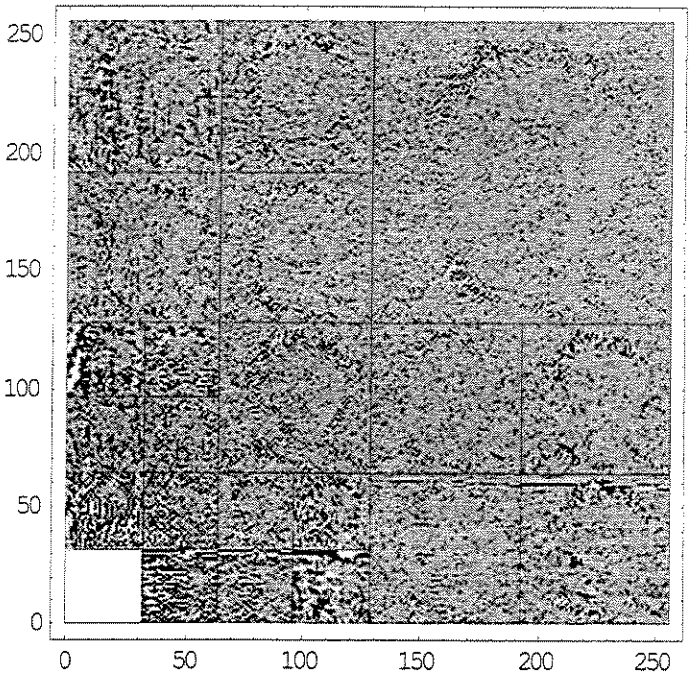
```
ShowBasisPosition2D[wpdata, AspectRatio -> 1]
```



- Graphics -

Note que os detalhes não estão separadamente normalizados . A área branca da esquinas mostra que tem muitos mais coeficientes que detalhes.

```
PlotCoefficients2D[wpdata]
```



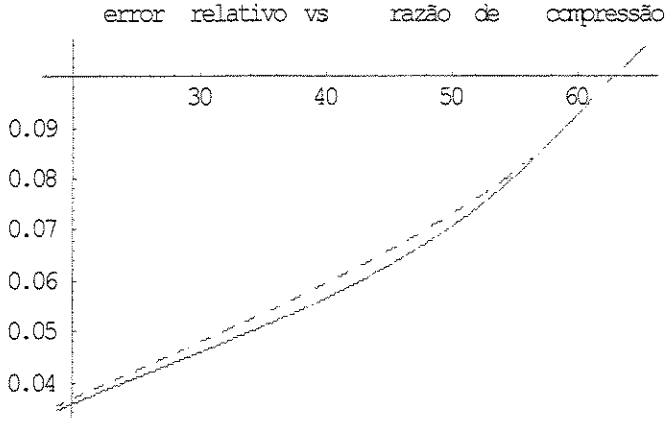
- Graphics -

Calculando a energia cumulativa, dos coeficiente da TW e de Wav. Packet

```
(wtE = CumulativeEnergy[wtdata, 3500] /
(Flatten[wtdata].Flatten[wtdata]);
wpE = CumulativeEnergy[wpdata, 3500] /
(Flatten[wpdata].Flatten[wpdata]));
```

Os coeficientes da Wav. Packet em linha continua,

```
Show[
{ListPlot[Transpose[{Table[256 256/i, {i, 1000, 3500}],
Sqrt[1 - Take[wtE, {1000, 3500}]]}], PlotJoined -> True,
DisplayFunction -> Identity, PlotStyle -> Dashing[{0.02}],
PlotLabel -> " error relativo vs. razão de compressão "],
ListPlot[Transpose[{Table[256 256/i, {i, 1000, 3500}],
Sqrt[1 - Take[wpE, {1000, 3500}]]}], PlotJoined -> True,
DisplayFunction -> Identity]},
DisplayFunction -> $DisplayFunction]
```



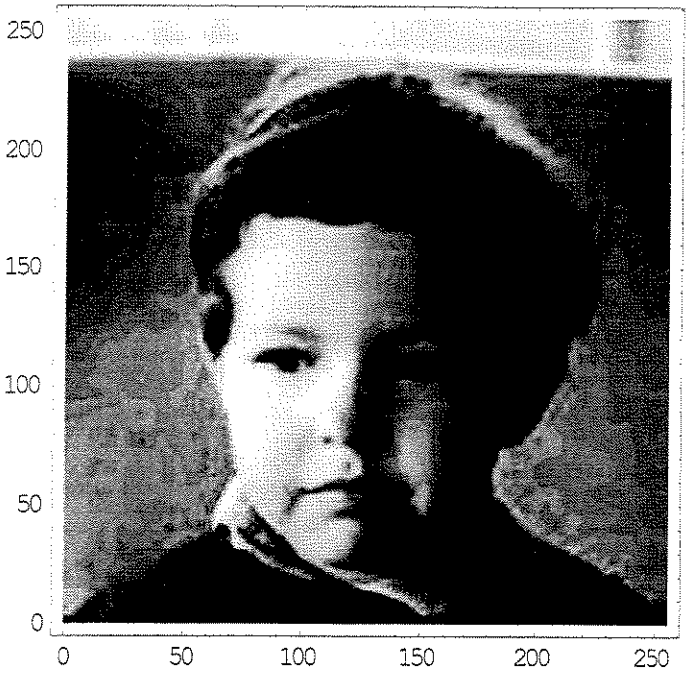
- Graphics -

Da curva vemos que se queremos manter o erro debaixo de 5%, a relação de compressão deveria estar abaixo aproximadamente 35. Nós escolhemos manter os 1985 coeficientes maiores. Isto corresponde a 3% do total coeficientes e para uma relação de compressão de aproximadamente 33.

```
InverseWaveletPacketTransform[
Compress[wpdata, 1985], s4,
BoundaryCondition -> Fixed];
```

Observe que o original é reproduzido bastante fielmente embora nós temos menos contraste que o original,

```
ListDensityPlot[%, Mesh -> False]
```



- DensityGraphics -

erro devido à compressão

```
Sqrt[#.# &[Flatten[data - %%]] /
(Flatten[data].Flatten[data])]
```

```
0.0506818
```

Erro usando a energia cumulativa

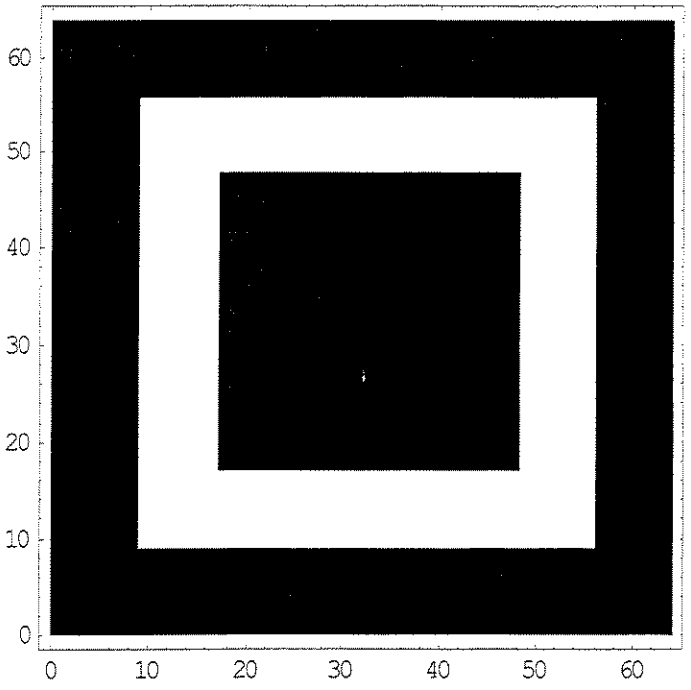
```
Sqrt[1 - wpE[[1985]]]
```

```
0.0487979
```

Outro exemplo com outro tipo de Imagem

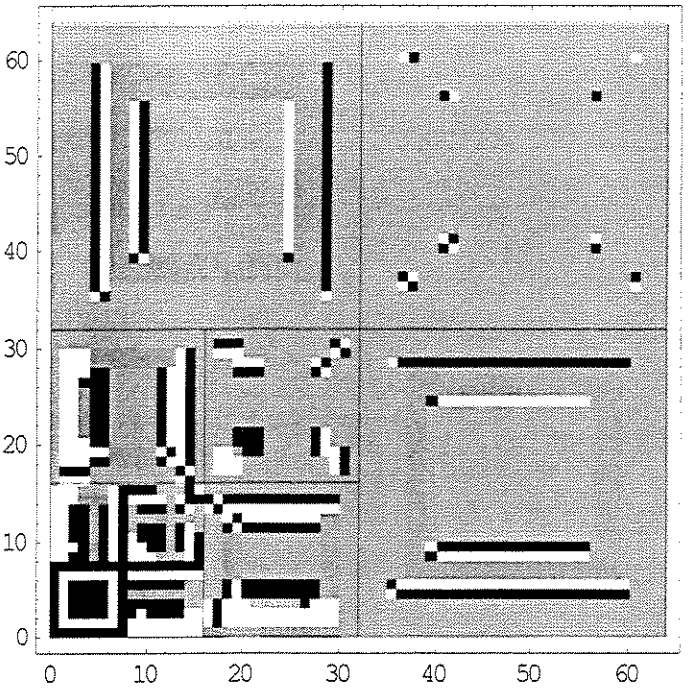
Função f(x, y) de tamanho {64,64}.

```
(f[x_, y_] := 1 /; (2 <= Abs[x] < 3 && Abs[y] < 3) || (
2 <= Abs[y] < 3 && Abs[x] < 3);
f[x_, y_] := 0;
data = Table[f[x, y], {x, -4, 4 - 1/8, 1/8},
{y, -4, 4 - 1/8, 1/8}];)
ListDensityPlot[data, Mesh -> False]
```



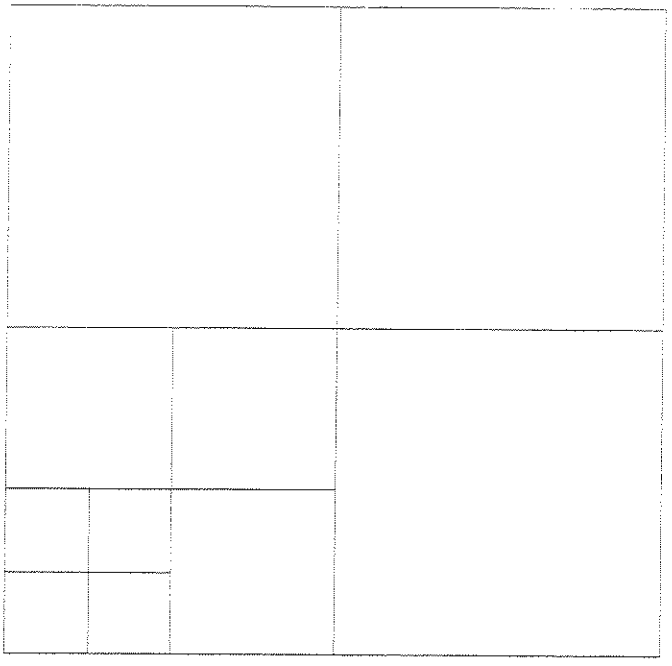
- DensityGraphics -

```
(d2 = DaubechiesFilter[2];
WaveletTransform[data, d2, 3];)
PlotCoefficients2D[%]
```



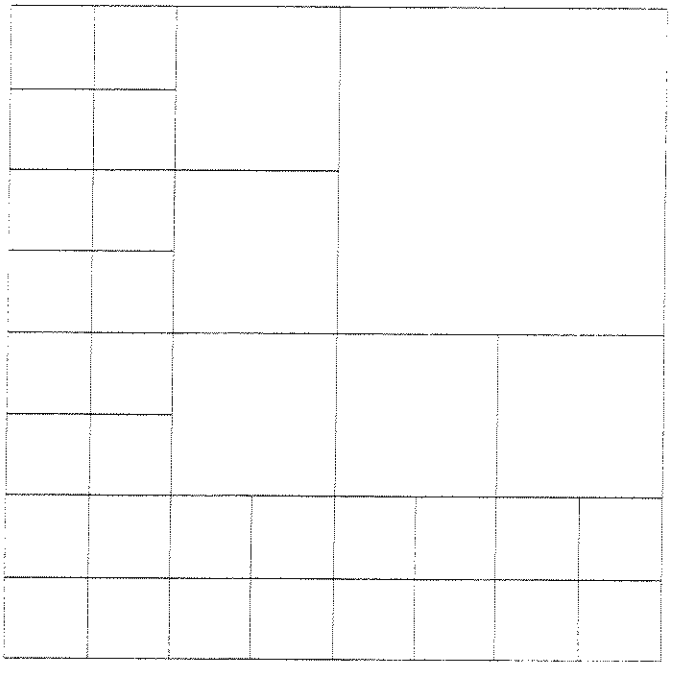
- Graphics -

```
ShowBasisPosition2D[%%, AspectRatio -> 1]
```



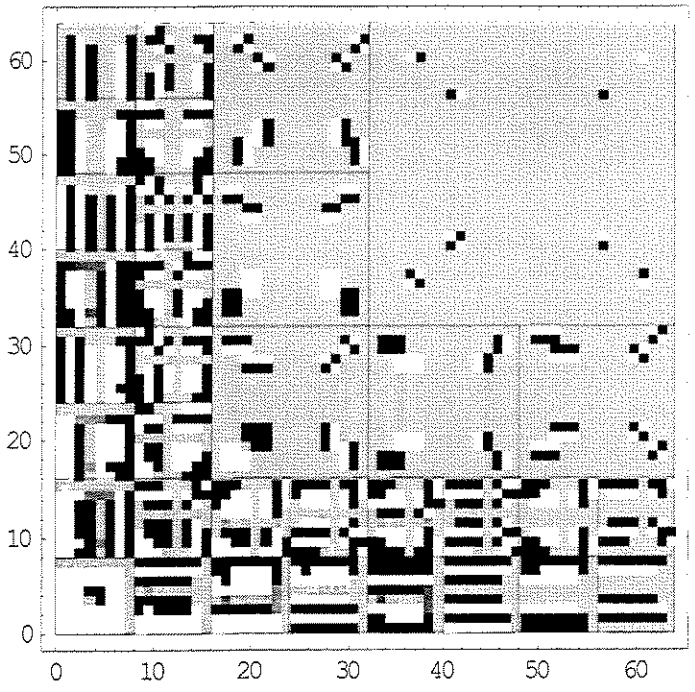
- Graphics -

```
WaveletPacketTransform[data, d2, 3,
BoundaryCondition -> Zero];
ShowBasisPosition2D[%, AspectRatio -> 1]
```



- Graphics -

```
PlotCoefficients2D[%%]
```



- Graphics -

ANEXO B.

PROCEDIMENTO *MARK_PARENTS* ("INFERIOR-SUPERIOR")

```
// File zerotree.cpp Zerotree class
#include "zerotree.h"
void tzerotree::mark_parents (int row,int col,short symbol) {
   // Marca os pais  com filhos significantes  (linha, coluna)
   // Os quartos filhos de posição  (i,j)  são: (2i-1,2j-1),(2i-1,2j),(2i,2j-1) e (2i,2j).
   int ichild = row,jchild = col;
   int iparent = ichild,jparent = jchild;
   while ((iparent > 1)&&(jparent > 1)) {
     if ((ichild/2)*2 == ichild) // even
          iparent = ichild/2;
     else // odd
          iparent = (ichild+1)/2;
     if ((jchild/2)*2 == jchild) // even
          jparent = jchild/2;
     else // odd
          jparent = (jchild+1)/2;
     set(iparent,jparent,symbol);
     ichild = iparent;
     jchild = jparent;
     } // end while
   return;
   }
```

PROCEDIMENTO *MARK_PARENTS* ("SUPERIOR-INFERIOR")

```
void tzerotree::mark_children (int row,int col,int nrows,
        int ncols,short symbol) {
   int child_end_row,child_end_col,child_rows,child_cols;
   // Os quartos filhos de posição  (i,j)  são: (2i-1,2j-1),(2i-1,2j),(2i,2j-1) e (2i,2j).
   child_end_row = 2*row;
   child_end_col = 2*col;
   child_rows = 2;
   child_cols = 2;
   while ((child_end_row<=nrows)&&(child_end_col<=ncols)) {
     for (int i=child_end_row-child_rows+1;i<=child_end_row;i++)
          for (int j=child_end_col-child_cols+1;j<=child_end_col;j++)
               set(i,j,symbol);
     child_end_row *= 2;
     child_end_col *= 2;
     child_rows *= 2;
     child_cols *= 2;
     } // end while
   return;
}
```

PRE-CODIFICADOR PACC

```
/*     SPLIT-PACC                                          */
/*Funções de preprocessamento do video */
/*Recebe uma sequencia de imagens e as precessas aplicando Transforma Wavelet*/
/*Hearders Padrao */
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <math.h>
```

```
#include <malloc.h>
#include <alloc.h>
/*Constantes*/
#define pi 3.14159265358979
#define raiz_2 1.41421356237
#define log_2  0.69314718 /*log_2 = ln*/
void frame_read(double **, char *, int , int );
void frame_write(double **, char* , int , int );
void  split_ex(double **, int, int, int);
/*************************************************************************
     This function READ a frame from a file
*************************************************************************/
 void frame_read(double **frame, char* nome, int V, int H)
{
 int i,j;
 FILE *fpin;
 /* Initialization of fread auxiliary vector ************************/
 unsigned char *row;
 row = new unsigned char [H];
 if ((fpin=fopen(nome,"rb")) == NULL)
                 {
                  printf("Unable to open read file %s\n", nome);
                  exit(1);
                 }
 for (i=0; i<V; i++)
    {
     fread(row, 1, H, fpin);
     for (j=0; j<H; j++)
        frame[i][j]=(double)row[j];
    }
 /* Destroying fread auxiliary vector ******************************/
 delete[] row;
 fclose(fpin);
}
/* This function copies frame_A into frame_B **********************/
void frame_copy(double **frame_B, double **frame_A, int V, int H)
{
 int i,j;
 for (i=0; i<V; i++)
    for (j=0; j<H; j++)
       frame_B[i][j]=frame_A[i][j];
}
/*************************************************************************
This function WRITE an unsigned char converted frame in a file
*************************************************************************/
void frame_write(double **frame, char nome[30], int V, int H)
{
 int i,j;
 FILE *fpout;
 /* Initialization of fread auxiliary vector ************************/
 unsigned char *row;
 row = new unsigned char [H];
 if ((fpout=fopen(nome,"wb")) == NULL)
                 {
                  printf("Unable to open file %s\n", nome);
                  exit(1);
                 }
 for (i=0; i<V; i++)
          {
        for (j=0; j<H; j++)
         if (frame[i][j]>=0 && frame[i][j]<=255)
           row[j]=(unsigned char)frame[i][j];
```

```
        else if (frame[i][j]>255)
              row[j]=255;
            else row[j]=0;
        fwrite(row, 1, H, fpout);
    }
 /* Destroying fwrite auxiliary vector ***************************/
        delete[] row;
        fclose(fpout);
}
/*************************************************************************
 This function WRITES a double matrix of NUM_COLUMS<=10 columns
 and length LENGTH<=100 into an ASCII file.
 *************************************************************************/
 void data_write(double **data, int LENGTH,  int NUM_COLUMNS, char nome[30])
                    {
                     int i,j;
    FILE *fpout;
                     if ((fpout=fopen(nome,"w")) == NULL)
                      {
                              printf("Unable to open the data file\n");
                              exit(1);
                              }
                    for (i=0; i<LENGTH; i++)
                    for (j=0; j<NUM_COLUMNS; j++)
                    if (j<NUM_COLUMNS-1)
                                        fprintf(fpout, "%f\t", data[i][j]);
                                        else fprintf(fpout, "%f\n",
                    data[i][NUM_COLUMNS-1]);

                    fclose(fpout);
                    }
/*************************************************************************
DESCOMPOSIÇAO
 This functions decomposes a frame into 4 subbands.
 The convolutions performed here with e_conv() only calculate half
 of the input points.
 Following the text notation, ML=p, MH=q, hl=bar g, hh=bar h
 *************************************************************************/
 /*************************************************************************
THE INPUT PARAMETERS ARE:
 **frame = the image to be decomposed, alocated as a bidimensional array ;
 V = vertical size ;
 H = horizontal size;
 ask_filter = the type of filter used (1,2)
 *************************************************************************/
 void  split_ex(double **frame, int V, int H, int ask_filter)
 {
 int i, j, ML, MH, i2, j2;
 void c_conv(double *, int, double *, int);
 int mod(int, int);
 void shift(double *, int, int);
 if (ask_filter==1)
   {
    ML=5;  /* Length of Lowband FIR analysis filter for QMF*/
    MH=3;  /* Length of Highband FIR analysis filter for QMF */
   }
 else if (ask_filter==2)
       {
        ML=9;/* Length of Lowband FIR analysis filter for VBL*/
        MH=7;/* Length of Highband FIR analysis filter for VBL */
       }
     else {
```

```
        printf("Choose 1 or 2 !!!\n");
        exit(1);
        }
/* Initialization and allocation of local filters*******************/
double *hl,*hh, *aux;
hl= new double [ML];
hh= new double [MH];
aux= new double [2*V-2];
/* Initialization and allocation of local frame matrixes***************/
      double **frame_aux, **frame_out;
/*double **frame_LL, **frame_LH, **frame_HH, **frame_HL;*/
frame_aux = new double *[2*V-2];
frame_out = new double *[V];
for (i=0; i<2*V-2; i++)
  {
   frame_aux[i]=new double [2*H-2];
   if (i<V)
     frame_out[i]=new double [H];
  }
/*********************************************************************
                                                Building the Filters: Choose one !
*********************************************************************/
 if (ask_filter==1)
  {
   /* QMF decomposition filters (5/3) */
   hl[0]=-0.125;
   hl[1]=0.25;
   hl[2]=0.75;
   hl[3]=0.25;
   hl[4]=-0.125;
   hh[0]=0.5;
   hh[1]=-1;
   hh[2]=0.5;
  }
else if (ask_filter==2)
     {
     /* Villasenor, Belzer & Liao best filter (9/7)*/
              hl[0]=0.037828/sqrt(2);
              hl[1]=-0.023849/sqrt(2);
              hl[2]=-0.110624/sqrt(2);
              hl[3]=0.377402/sqrt(2);
              hl[4]=0.852699/sqrt(2);
              hl[5]=0.377402/sqrt(2);
              hl[6]=-0.110624/sqrt(2);
              hl[7]=-0.023849/sqrt(2);
              hl[8]=0.037828/sqrt(2);
              hh[0]=0.064359*sqrt(2);
              hh[1]=-0.040689*sqrt(2);
              hh[2]=-0.418092*sqrt(2);
              hh[3]=0.788486*sqrt(2);
              hh[4]=-0.418092*sqrt(2);
              hh[5]=-0.040689*sqrt(2);
              hh[6]=0.064359*sqrt(2);
     }
/* Spline decomposition Filters l=3, k=4, k~=2 *
/*
hl[0]=sqrt(2)*0.02343750000000;
hl[1]=-sqrt(2)*0.04687500000000;
hl[2]=-sqrt(2)*0.12500000000000;
hl[3]=sqrt(2)*0.29687500000000;
hl[4]=sqrt(2)*0.70312500000000;
hl[5]=sqrt(2)*0.29687500000000;
```

```
hl[6]=-sqrt(2)*0.12500000000000;
hl[7]=-sqrt(2)*0.04687500000000;
hl[8]=sqrt(2)*0.02343750000000;
hh[0]=-sqrt(2)*0.25;
hh[1]=sqrt(2)*0.5;
hh[2]=-sqrt(2)*0.25;
*/
/* Biorthogonal 1.5 wavelet decomposition filters*/
/*
hl[0]=0.01657281518406;
hl[1]=-0.01657281518406;
hl[2]=-0.12153397801644;
hl[3]=0.12153397801644;
hl[4]=0.70710678118655;
hl[5]=0.70710678118655;
hl[6]=0.12153397801644;
hl[7]=-0.12153397801644;
hl[8]=-0.01657281518406;
hl[9]=0.01657281518406;
hh[0]=0;
hh[1]=0;
hh[2]=0;
hh[3]=0;
hh[4]=-0.70710678118655;
hh[5]=0.70710678118655;
hh[6]=0;
hh[7]=0;
hh[8]=0;
hh[9]=0;
*/
/* Biorthogonal 3.3 wavelet decomposition filters
*/
/*
hl[0]=0.06629126073624;
hl[1]=-0.19887378220872;
hl[2]=-0.15467960838456;
hl[3]=0.99436891104358;
hl[4]=0.99436891104358;
hl[5]=-0.15467960838456;
hl[6]=-0.19887378220872;
hl[7]=0.06629126073624;
hh[0]=0;
hh[1]=0;
hh[2]=-0.17677669529664;
hh[3]=0.53033008588991;
hh[4]=-0.53033008588991;
hh[5]=0.17677669529664;
hh[6]=0;
hh[7]=0;
*/
/*********************************************************************
                                        LL Band
*********************************************************************
      Convolution of the lines with the filter and writing to frame_aux
*********************************************************************/
for (i=0; i<V; i++)
   for (j=0; j<2*H-2; j++)
      {
       if (j>=H)
         j2=2*H-j-2;
       else
         j2=j;
```

```
      frame_aux[i][j]=frame[i][j2];
     }
 for (i=0; i<V; i++)
   {
    c_conv(frame_aux[i],2*H-2,hl,ML);
    if (mod((ML-1)/2,2)==0)
      shift(frame_aux[i], 2*H-2, -(ML-1)/2);
    else
      shift(frame_aux[i], 2*H-2, -(ML-1)/2-1);
   }
/*********************************************************************
Convolution of the collumns with the filter and writing to frame_aux
*********************************************************************/
for (j=0; j<H; j+=2)
   {
    for (i=0; i<2*V-2; i++)
       {
        if (i>=V)
          i2=2*V-i-2;
        else
          i2=i;
        aux[i]=frame_aux[i2][j];
       }
    c_conv(aux,2*V-2,hl,ML);
    if (mod((ML-1)/2,2)==0)
      shift(aux, 2*V-2, -(ML-1)/2);
    else
      shift(aux, 2*V-2, -(ML-1)/2-1);

    for (i=0; i<V; i++)
       frame_aux[i][j]=aux[i];
   }
/**Saving in frame_out************************************************/
 for (i=0; i<V; i+=2)
   for (j=0; j<H; j+=2)
     frame_out[i/2][j/2]=frame_aux[i][j];
     /*frame_LL[i][j]=frame_out[i/2][j/2]; */
/*********************************************************************
                                                LH band
*********************************************************************
Convolution of the lines with the filter and writing to frame_out
*********************************************************************/
for (i=0; i<V; i++)
   for (j=0; j<2*H-2; j++)
      {
       if (j>=H)
         j2=2*H-j-2;
       else
         j2=j;
       frame_aux[i][j]=frame[i][j2];
      }
for (i=0; i<V; i++)
   {
    c_conv(frame_aux[i],2*H-2,hl,ML);
    if (mod((ML-1)/2,2)==0)
      shift(frame_aux[i], 2*H-2, -(ML-1)/2);
    else
      shift(frame_aux[i], 2*H-2, -(ML-1)/2-1);
   }
/*********************************************************************
Convolution of the collumns with the filter and writing to frame_aux
*********************************************************************/
```

```
for (j=0; j<H; j+=2)
   {
    for (i=0; i<2*V-2; i++)
       {
        if (i>=V)
          i2=2*V-i-2;
        else
          i2=i;
        aux[i]=frame_aux[i2][j];
       }
    c_conv(aux,2*V-2,hh,MH);
    if (mod((MH-1)/2,2)==0)
      shift(aux, 2*V-2, -(MH-1)/2);
    else
      shift(aux, 2*V-2, -(MH-1)/2-1);
    for (i=0; i<V; i++)
       frame_aux[i][j]=aux[i];
   }
/* Saving in frame_out ************************************************/
for (i=0; i<V; i+=2)
   for (j=0; j<H; j+=2)
      frame_out[V/2+i/2][j/2]=frame_aux[i][j];
      /*frame_LH[i][j]=frame_out[V/2+i/2][j/2];*/
 /*****************************************************************************
                                    HL band
*****************************************************************************
       Convolution of the lines with the filter and writing to frame_out
*****************************************************************************/
 for (i=0; i<V; i++)
   for (j=0; j<2*H-2; j++)
      {
       if (j>=H)
         j2=2*H-j-2;
       else
         j2=j;
      frame_aux[i][j]=frame[i][j2];
      }
for (i=0; i<V; i++)
   {
    c_conv(frame_aux[i],2*H-2,hh,MH);
    if (mod((MH-1)/2,2)==0)
      shift(frame_aux[i], 2*H-2, -(MH-1)/2);
    else
      shift(frame_aux[i], 2*H-2, -(MH-1)/2-1);
   }
/*****************************************************************
Convolution of the collumns with the filter and writing to frame_aux
*****************************************************************************/
for (j=0; j<H; j+=2)
   {
    for (i=0; i<2*V-2; i++)
       {
        if (i>=V)
          i2=2*V-i-2;
        else
          i2=i;
        aux[i]=frame_aux[i2][j];
       }
    c_conv(aux,2*V-2,hl,ML);
    if (mod((ML-1)/2,2)==0)
      shift(aux, 2*V-2, -(ML-1)/2);
   else
```

```
      shift(aux, 2*V-2, -(ML-1)/2-1);

    for (i=0; i<V; i++)
       frame_aux[i][j]=aux[i];
  }
/* Saving in frame_out ************************************************/
for (i=0; i<V; i+=2)
   for (j=0; j<H; j+=2)
      frame_out[i/2][H/2+j/2]=frame_aux[i][j];
      /*frame_HL[i][j]=frame_out[i/2][H/2+j/2];*/
 /*********************************************************************
                                                    HH band
 *********************************************************************
  Convolution of the lines with the filter and writing to frame_out
 *********************************************************************/
for (i=0; i<V; i++)
   for (j=0; j<2*H-2; j++)
      {
       if (j>=H)
         j2=2*H-j-2;
       else
         j2=j;
       frame_aux[i][j]=frame[i][j2];
      }
for (i=0; i<V; i++)
   {
    c_conv(frame_aux[i],2*H-2,hh,MH);
    if (mod((MH-1)/2,2)==0)
      shift(frame_aux[i], 2*H-2, -(MH-1)/2);
    else
      shift(frame_aux[i], 2*H-2, -(MH-1)/2-1);
   }
/*********************************************************************
Convolution of the collumns with the filter and writing to frame_aux
*********************************************************************/
for (j=0; j<H; j+=2)
   {
    for (i=0; i<2*V-2; i++)
       {
        if (i>=V)
          i2=2*V-i-2;
        else
          i2=i;
        aux[i]=frame_aux[i2][j];
       }
    c_conv(aux,2*V-2,hh,MH);
    if (mod((MH-1)/2,2)==0)
      shift(aux, 2*V-2, -(MH-1)/2);
    else
      shift(aux, 2*V-2, -(MH-1)/2-1);

    for (i=0; i<V; i++)
       frame_aux[i][j]=aux[i];
  }
/* Saving in frame_out ************************************************/
                for (i=0; i<V; i+=2)
                for (j=0; j<H; j+=2)
                frame_out[V/2+i/2][H/2+j/2]=frame_aux[i][j];
  /*frame_HH[i][j]=frame_out[V/2+i/2][H/2+j/2]; */
/*********************************************************************
                                          Saving frame_out in frame
        frame[i][j] are all the frame_out
```

```
  frame_LL= all the frame LL, frame_LH= all the frame LH, idem for HL and HH
  ***************************************************************************/
  for (i=0; i<V; i++)
     for (j=0; j<H; j++)
        frame[i][j]=frame_out[i][j];
  for (i=0; i<2*V-2; i++)
     {
      delete[] frame_aux[i];
      if (i<V)
      delete[] frame_out[i];
     }
  delete[] hl, hh, aux, frame_aux, frame_out;
}
/***************************************************************************
   This function returns the positive integer a modulus b
****************************************************************************/
                int mod(int a, int b)
                {
                if (a>=0)
                        return(a-(a/b)*b);
        else return mod(b-mod(-a,b),b);
                }
/***************************************************************************
THE CONVOLUTION: CHOOSE  ONE !!
                                                        and
            Choose Alternative I, II or III !!!
****************************************************************************/
/***************************************************************************
 This function circularly or symetricaly convolves x of size N with h
 of size M and (M<N) and saves the result in x
****************************************************************************/
                void c_conv(double *x, int N, double *h, int M)
   {
                                int n, k;
                                double *aux_x;
                                aux_x = new double [N];

  /* Copying x to an auxiliary vector ************************************/
                                for (n=0; n<N; n++)
                        {
                        aux_x[n]=x[n];
                        x[n]=0;
                        }
  /***************************************************************************
                                        Choose Alternative I, II or III !!!
  ****************************************************************************/
  /* Alternative I - One way of circular convolution
                                                        Only close to the first edge
************************/

                        /*
                                for (n=0; n<M; n++)
          for (k=0; k<M; k++)
             x[n]+=h[k]*aux_x[mod(n-k,N)];
  */
  /* Alternative II - Another way of circular convolution
            Only close to the first edge "Might be faster" than I */
                for (n=0; n<M; n++)
                {
                 for (k=0; k<=n; k++)
        x[n]+=h[k]*aux_x[n-k];
                for (k=n+1; k<M; k++)
```

```
                        x[n]+=h[k]*aux_x[N+(n-k)];
                    }
                    for (n=M; n<N; n++)
          for (k=0; k<M; k++)
                     x[n]+=h[k]*aux_x[n-k];
 delete[] aux_x;
}
/*************************************************************************
This function circularly or symetricaly convolves x of size N
with h of size M and (M<N) and saves the "even points" in x
                                                    It's INTENDED TO BE USED IN  split().
*************************************************************************/
                    void ev_conv(double *x, int N, double *h, int M)
                    {
                    int n, k;
                    double *aux_x;
                    aux_x = new double [N];
/* Copying x to an auxiliary vector ***************************************/
                    for (n=0; n<N; n++)
                    {
                    aux_x[n]=x[n];
                    x[n]=0;
                    }
 /************************************************************************
                              Choose Alternative I or II !!!
 ************************************************************************/
 /* Alternative I - One way of circular convolution
Only close to the first edge
*************************/
 /*
 for (n=0; n<M; n++)
    for (k=0; k<M; k++)
       x[n]+=h[k]*aux_x[mod(n-k,N)];
 */
 /* Alternative II - Another way of circular convolution
             Only close to the first edge "Might be faster" than I */
                    for (n=0; n<M; n+=2)
                    {
                    for (k=0; k<=n; k++)
          x[n]+=h[k]*aux_x[n-k];
                    for (k=n+1; k<M; k++)
                      x[n]+=h[k]*aux_x[N+(n-k)];
                    }
 /************************************************************************
 If Alternatives I or II were chosen, the next paragraph must be used
 ************************************************************************/
                    for (n=M+mod(M,2); n<N; n+=2)
                    for (k=0; k<M; k++)
                      x[n]+=h[k]*aux_x[n-k];
                    delete[] aux_x;
                    }
/*************************************************************************
                              This function performs a circular shift on an array
*************************************************************************/
                   void shift(double *array, int LENGTH, int H_DELAY)
                   {
                    double *aux;
    aux = new double [LENGTH];
                    int i;
                    for (i=0; i<LENGTH; i++)
                   aux[i]=array[i];
                    for (i=0; i<LENGTH; i++)
```

```
                array[i]=aux[mod(i-H_DELAY, LENGTH)];
                delete[] aux;
                }
/*************************************************************************
This function symetrically completes an array around the CENTER (F_LENGTH-1)/2
*************************************************************************/
                void complete(double *array, int LENGTH, int F_LENGTH)
                        {
      int i,L;
                         L=(LENGTH+2)/2;
                        for (i=(F_LENGTH-1)/2-1; i>=(F_LENGTH-1)/2-(L-2); i--)
       array[mod(i, LENGTH)]=array[(F_LENGTH-1)/2+(F_LENGTH-1)/2-i];
                        }
/*************************************************************************
                                                        STATIStic do Frame
*************************************************************************
 ************************************************************************
 This function returns the min or max value contained in frame,
                                                        depending on ask
 ************************************************************************/
                double info(double **frame, int V, int H, char ask[3])
                {
    int i,j;
                 double min, max;
    max=0;
                 min=1000;
                for (i=0; i<V; i++)
       for (j=0; j<H; j++)
        {
                if (frame[i][j]<min)
                        min=frame[i][j];
                if (frame[i][j]>max)
                        max=frame[i][j];
        }
                if (!strcmp(ask,"MIN"))
                 return min;
                else return max;
                }
 /************************************************************************
                        This function returns the MEAN of a frame
 ************************************************************************/
                double frame_mean(double **frame, int V, int H)
                {
                 int i,j;
                 double sum;

                 sum=0;
                 for (i=0; i<V; i++)
      for (j=0; j<H; j++)
          sum+=frame[i][j];
                return sum/(V*H);
                }
 /************************************************************************
 This function adds a constant "CONST" to all the elements of the frame
 ************************************************************************/
                void frame_add_const(double **frame, int V, int H, double CONST)
                {
                 int i,j;

                for (i=0; i<V; i++)
                for (j=0; j<H; j++)
                 frame[i][j]+=CONST;
```

```
}
/*************************************************************************
This function multiplies a constant "CONST" to all the elements of the frame
*************************************************************************/
                void frame_mul_const(double **frame, int V, int H, double CONST)
                {
                 int i,j;
                for (i=0; i<V; i++)
        for (j=0; j<H; j++)
                 frame[i][j]*=CONST;
                }
/*********************************************************************
This function returns the Standard Deviation of a Zero Mean frame
*********************************************************************/
                double frame_std(double **frame, int V, int H)
                {
                 int i,j;
                 double sum, mean;
                 sum=0;
                mean=frame_mean(frame, V, H);
                frame_add_const(frame, V, H, -mean);
                for (i=0; i<V; i++)
        for (j=0; j<H; j++)
      sum+=frame[i][j]*frame[i][j];
                frame_add_const(frame, V, H, mean);
                return sqrt(sum/(V*H));
                }
/*********************************************************************
This function takes the Square Root of the frame
*************************************************************************/
                void frame_sqrt(double **frame, int V, int H)
                {
                 int i,j;
                for (i=0; i<V; i++)
                for (j=0; j<H; j++)
       frame[i][j]=sqrt(frame[i][j]);
                }

/*************************************************************************
This function squares the frame to a Power of 2
*************************************************************************/
                void frame_sqr(double **frame, int V, int H)
                {
                 int i,j;
                for (i=0; i<V; i++)
                for (j=0; j<H; j++)
       frame[i][j]=pow(frame[i][j],2);
                }
/*************************************************************************
                RECONSTRUÇAO
This code builds a frame from 4 subbands.
The convolutions performed  here with c_conv() are commom circular
convolutions calculating the   total of input points.
Following the text notation, ML=q, MH=p, gl=g, gh=h
*************************************************************************/
/*************************************************************************
The input parameters are :
**frame = the decomposed image, alocated as a
bidimensional array ;
V = vertical size ;
H = horizontal size;
ask_filter = the type of filter used (1,2)
```

```
*****************************************************************************/
                void build_ex(double **frame, int V, int H, int ask_filter)
                {
        int i, j, ML, MH;
                if (ask_filter==1)
                {
        ML=3;  /* Length of Lowband FIR Synthesis filter for QMF*/
        MH=5;  /* Length of Highband FIR Synthesis filter for QMF */
                }
                                        else if (ask_filter==2)
        {
       ML=7;/* Length of Lowband FIR Synthesis filter for VBL*/
       MH=9;/* Length of Highband FIR Synthesis filter for VBL */
        }
                else {
        printf("Choose 1 or 2 !!!\n");
        exit(1);
                }
/*Initialization and allocation of local filters*********************/
 /* These are the synthesis ones */
 double *gl,*gh, *aux;
 gl= new double [ML];
 gh= new double [MH];
 aux= new double [2*V-2];
 /* Initialization and allocation of local frame matrixes***************/
 double **frame_aux, **frame_out;
 frame_aux = new double *[2*V-2];
 frame_out = new double *[V];
 for (i=0; i<2*V-2; i++)
   {
    frame_aux[i]=new double [2*H-2];
    if (i<V)
      frame_out[i]=new double [H];
   }
/*****************************************************************************
Building the filters Choose one !
*****************************************************************************/
 if (ask_filter==1)
  {
  /* QFM synthesis filters (5/3)*/
       gl[0]=0.5;
    gl[1]=1;
    gl[2]=0.5;
    gh[0]=0.125;
    gh[1]=0.25;
    gh[2]=-0.75;
    gh[3]=0.25;
    gh[4]=0.125;
  }
 else if (ask_filter==2)
       {
       /* Villasenor, Belzer & Liao best filter (9/7)*/
        gl[0]=-0.064359*sqrt(2);
        gl[1]=-0.040689*sqrt(2);
        gl[2]=0.418092*sqrt(2);
        gl[3]=0.788486*sqrt(2);
        gl[4]=0.418092*sqrt(2);
        gl[5]=-0.040689*sqrt(2);
        gl[6]=-0.064359*sqrt(2);
        gh[0]=0.037828/sqrt(2);
        gh[1]=0.023849/sqrt(2);
        gh[2]=-0.110624/sqrt(2);
```

```
        gh[3]=-0.377402/sqrt(2);
        gh[4]=0.852699/sqrt(2);
        gh[5]=-0.377402/sqrt(2);
        gh[6]=-0.110624/sqrt(2);
        gh[7]=0.023849/sqrt(2);
        gh[8]=0.037828/sqrt(2);
       }
/* Spline Synthesis Filters l=3, k=4, k~=2 */
/*
gl[0]=sqrt(2)*0.25;
gl[1]=sqrt(2)*0.5;
gl[2]=sqrt(2)*0.25;
gh[0]=sqrt(2)*0.02343750000000;
gh[1]=sqrt(2)*0.04687500000000;
gh[2]=-sqrt(2)*0.12500000000000;
gh[3]=-sqrt(2)*0.29687500000000;
gh[4]=sqrt(2)*0.70312500000000;
gh[5]=-sqrt(2)*0.29687500000000;
gh[6]=-sqrt(2)*0.12500000000000;
gh[7]=sqrt(2)*0.04687500000000;
gh[8]=sqrt(2)*0.02343750000000;
*/
/* Biorthogonal 1.5 wavelet synthesis filters */
/*
gl[0]=0;
gl[1]=0;
gl[2]=0;
gl[3]=0;
gl[4]=0.70710678118655;
gl[5]=0.70710678118655;
gl[6]=0;
gl[7]=0;
gl[8]=0;
gl[9]=0;
gh[0]=0.01657281518406;
gh[1]=0.01657281518406;
gh[2]=-0.12153397801644;
gh[3]=-0.12153397801644;
gh[4]=0.70710678118655;
gh[5]=-0.70710678118655;
gh[6]=0.12153397801644;
gh[7]=0.12153397801644;
gh[8]=-0.01657281518406;
gh[9]=-0.01657281518406;
*/
/* Biorthogonal 3.3 wavelet synthesis filters */
/*
gl[0]=0;
gl[1]=0;
gl[2]=0.17677669529664;
gl[3]=0.53033008588991;
gl[4]=0.53033008588991;
gl[5]=0.17677669529664;
gl[6]=0;
gl[7]=0;
gh[0]=0.06629126073624;
gh[1]=0.19887378220872;
gh[2]=-0.15467960838456;
gh[3]=-0.99436891104358;
gh[4]=0.99436891104358;
gh[5]=0.15467960838456;
gh[6]=-0.19887378220872;
```

```
gh[7]=-0.06629126073624;
*/
/* Making frame_aux and frame_out null ************************************/
                for (i=0; i<2*V-2; i++)
                for (j=0; j<2*H-2; j++)
                {
                frame_aux[i][j]=0;
                if ((i<V) && (j<H))
                        frame_out[i][j]=0;
                }
/*********************************************************************
                                                                    LL Band
*********************************************************************/
                for (i=0; i<V; i+=2)
                for (j=0; j<H; j+=2)
                frame_aux[i][j]=frame[i/2][j/2];
/*********************************************************************
Convolution of the collumns with the filter and writing to frame_aux
*********************************************************************/
                for (j=0; j<H; j+=2)
                {
                for (i=0; i<2*V-2; i++)
                aux[i]=frame_aux[i][j];
                if (mod((MH-1)/2,2)==0)
                shift(aux, 2*V-2, (MH-1)/2);
                else
                shift(aux, 2*V-2, (MH-1)/2+1);
                complete(aux, 2*V-2, MH);
                c_conv(aux,2*V-2,gl,ML);
                shift(aux, 2*V-2, -(ML-1)/2-(MH-1)/2);
                for (i=0; i<V; i++)
                frame_aux[i][j]=aux[i];
                }
/*********************************************************************
Convolution of the lines with the filter and writing to frame_aux
*********************************************************************/
                for (i=0; i<V; i++)
                {
                if (mod((MH-1)/2,2)==0)
                shift(frame_aux[i], 2*H-2, (MH-1)/2);
                else
                shift(frame_aux[i], 2*H-2, (MH-1)/2+1);
                complete(frame_aux[i], 2*H-2, MH);
                c_conv(frame_aux[i], 2*H-2, gl, ML);
                shift(frame_aux[i], 2*H-2, -(ML-1)/2-(MH-1)/2);
                }
/* Saving in frame_out ************************************************/
                for (i=0; i<V; i++)
                for (j=0; j<H; j++)
                frame_out[i][j]+=frame_aux[i][j];
/* Making frame_aux null **********************************************/
                for (i=0; i<2*V-2; i++)
                for (j=0; j<2*H-2; j++)
                frame_aux[i][j]=0;
/*********************************************************************
                                                                    LH Band
*********************************************************************/
                for (i=0; i<V; i+=2)
                for (j=0; j<H; j+=2)
        frame_aux[i][j]=frame[i/2+V/2][j/2];
/*********************************************************************
Convolution of the collumns with the filter and writing to frame_aux
```

```
**********************************************************************/
                for (j=0; j<H; j+=2)
                {
                for (i=0; i<2*V-2; i++)
                aux[i]=frame_aux[i][j];
                if (mod((ML-1)/2,2)==0)
                shift(aux, 2*V-2, (ML-1)/2);
                else
                shift(aux, 2*V-2, (ML-1)/2+1);
                complete(aux, 2*V-2, ML);
                c_conv(aux,2*V-2,gh,MH);
                shift(aux, 2*V-2, -(ML-1)/2-(MH-1)/2);
                for (i=0; i<V; i++)
                frame_aux[i][j]=aux[i];
                }
/**********************************************************************
Convolution of the lines with the filter and writing to frame_aux
**********************************************************************/
                for (i=0; i<V; i++)
                {
                if (mod((MH-1)/2,2)==0)
                shift(frame_aux[i], 2*H-2, (MH-1)/2);
                else
                shift(frame_aux[i], 2*H-2, (MH-1)/2+1);
                complete(frame_aux[i], 2*H-2, MH);
                c_conv(frame_aux[i],2*H-2,gl,ML);
                shift(frame_aux[i], 2*H-2, -(ML-1)/2-(MH-1)/2);
                }
/* Saving in frame_out ***********************************************/
                for (i=0; i<V; i++)
                for (j=0; j<H; j++)
                frame_out[i][j]+=frame_aux[i][j];
/* Making frame_aux null *********************************************/
                for (i=0; i<2*V-2; i++)
                for (j=0; j<2*H-2; j++)
                frame_aux[i][j]=0;
/**********************************************************************
                                                                    HL Band
**********************************************************************/
                for (i=0; i<V; i+=2)
                for (j=0; j<H; j+=2)
                frame_aux[i][j]=frame[i/2][j/2+H/2];
/**********************************************************************
Convolution of the collumns with the filter and writing to frame_aux
**********************************************************************/
                for (j=0; j<H; j+=2)
                {
                for (i=0; i<2*V-2; i++)
                aux[i]=frame_aux[i][j];
                if (mod((MH-1)/2,2)==0)
                shift(aux, 2*V-2, (MH-1)/2);
                else
                shift(aux, 2*V-2, (MH-1)/2+1);
                complete(aux, 2*V-2, MH);
                c_conv(aux,2*V-2,gl,ML);
                shift(aux, 2*V-2, -(ML-1)/2-(MH-1)/2);
                for (i=0; i<V; i++)
                frame_aux[i][j]=aux[i];
                }
/**********************************************************************
Convolution of the lines with the filter and writing to frame_aux
**********************************************************************/
```

```
                for (i=0; i<V; i++)
                {
                if (mod((ML-1)/2,2)==0)
                shift(frame_aux[i], 2*H-2, (ML-1)/2);
                else
        shift(frame_aux[i], 2*H-2, (ML-1)/2+1);
                complete(frame_aux[i], 2*H-2, ML);
                c_conv(frame_aux[i],2*H-2,gh,MH);
                shift(frame_aux[i], 2*H-2, -(ML-1)/2-(MH-1)/2);
                }
/* Saving in frame_out ************************************************/
                for (i=0; i<V; i++)
                for (j=0; j<H; j++)
                frame_out[i][j]+=frame_aux[i][j];
/* Making frame_aux null **********************************************/
for (i=0; i<2*V-2; i++)
   for (j=0; j<2*H-2; j++)
      frame_aux[i][j]=0;
/*************************************************************************
                                                                    HH Band
*************************************************************************/
                for (i=0; i<V; i+=2)
                for (j=0; j<H; j+=2)
                frame_aux[i][j]=frame[i/2+V/2][j/2+H/2];
/*************************************************************************
Convolution of the collumns with the filter and writing to frame_aux
*************************************************************************/
                for (j=0; j<H; j+=2)
                {
                for (i=0; i<2*V-2; i++)
                aux[i]=frame_aux[i][j];
                if (mod((ML-1)/2,2)==0)
                shift(aux, 2*V-2, (ML-1)/2);
                else
                shift(aux, 2*V-2, (ML-1)/2+1);
                        complete(aux, 2*V-2, ML);
                        c_conv(aux,2*V-2,gh,MH);
                        shift(aux, 2*V-2, -(ML-1)/2-(MH-1)/2);
                for (i=0; i<V; i++)
                frame_aux[i][j]=aux[i];
                }
/*************************************************************************
Convolution of the lines with the filter and writing to frame_aux
*************************************************************************/
                for (i=0; i<V; i++)
                {
                if (mod((ML-1)/2,2)==0)
                shift(frame_aux[i], 2*H-2, (ML-1)/2);
                else
                shift(frame_aux[i], 2*H-2, (ML-1)/2+1);
                        complete(frame_aux[i], 2*H-2, ML);
                        c_conv(frame_aux[i],2*H-2,gh,MH);
                        shift(frame_aux[i], 2*H-2, -(ML-1)/2-(MH-1)/2);
                }
/* Saving in frame_out ************************************************/
                for (i=0; i<V; i++)
                for (j=0; j<H; j++)
                frame_out[i][j]+=frame_aux[i][j];
/* Saving frame_out in frame ******************************************/
                for (i=0; i<V; i++)
                for (j=0; j<H; j++)
                frame[i][j]=frame_out[i][j];
```

```
                    for (i=0; i<2*V-2; i++)
                    {
                    delete[] frame_aux[i];
                    if (i<V)
                    delete[] frame_out[i];
                    }
                    delete[] gl, gh, aux, frame_aux, frame_out;
                    }
   /*Alocaçao da sequencia de Imagens*************************************/
  /*int V= 512;
  int H=512;
  int ask_filter= 2; /*Colocar opçao de preguntar 1 ou 2*/
  /*int i,j;  */
  /*double **frame;
    frame = new double *[V];
  for (i=0; i<V; i++)
     {
      frame[i]=new double [H];
     } */
   /* double **frame_1, **frame_2;
     frame_1 = (double **)calloc(V,sizeof(double * ));
     frame_2 = (double **)calloc(V,sizeof(double * ));
                    for (i=0; i<V; i++)
    {
          frame_1[i]=(double *)calloc(H,sizeof(double));
          frame_2[i]=(double *)calloc(H,sizeof(double));
    } */
    /*Outra forma de alocar*************************************
    double **frame_1, **frame_2;
  frame_1 = new double *[V];
  frame_2 = new double *[V];
  for (i=0; i<V; i++)
     {
     frame_1[i]=new double [H];
     frame_2[i]=new double [H];
     }   */
void main ()
{
     int V, H,i;
          int ask_filter;
     double **frame;
     char nome_int[30];
     char nome_out[30];
    /*********************************************************************/
  printf("É»                                                                  ");
  printf("º Split01 - Sistema para Pre_processamento de Video com TW (Coder)  º");
  printf("º LCV- Lab. de Comunic. Visuais                                     º");
  printf("ºDECOM/FEEC -Departamento de Comunic./Fac. Eng. Eletrica e Comp.    º");
  printf("º Unicamp    - Universidade Estadual de Campinas                    º");
  printf("º Copyrigth (C) - 2001 por Vicente Becerra & Yuzo Iano              º");
  printf("º Versão Beta.0                                                     º");
  printf("È                                                                   ");
    /*********************************************************************
              Read data  from the file
    *********************************************************************/
     printf("\n Digite o nome do arquivo_imagem a ser codificada: ");
     scanf("%s",nome_int);
     printf("\n Digite o nome do arquivo_imagem a ser almazenada: ");
     scanf("%s",nome_out);
     printf("\n Digite o valor de V : ");
     scanf("%d",&V);
     printf("\n Digite o valor de H : ");
```

```
 scanf("%d",&H);
/*********************************************************************
                Allocation Frame and  Read a Frame from a file
*********************************************************************/
                        if( (frame = (double**)calloc(V,sizeof(double*)) )!= NULL ){
    for (i=0; i<V; i++)
                        {
                                if( ( frame[i]=(double*)calloc(H,sizeof(double)) )==NULL ){
                printf("Nao pude alocar memoria 2\n");
          exit(0);
        }
                        }
                }else{
    printf("Nao pude alocar memoria 1\n");
    exit(0);
    }
  /*frame = (double **)calloc(V,sizeof(double * ));
                for (i=0; i<V; i++)
                                        {
                                frame[i]=(double *)calloc(H,sizeof(double));
                        } */
    frame_read(frame,nome_int,V,H);
 /*************************************************************
 Descomposes a frame in 3 niv.
 Record frame_LL (em cada subband )
 *************************************************************/
 /*  Descomposes a frame:4 subbandas
    Record frame_LL
 *************************************************************/
                        printf("\n Selecione o tipo de filtro 1 ou 2: ");
  scanf("%d",&ask_filter);

  split_ex (frame,V,H,ask_filter);
   /*********************************************************
    Descomposes a frame:7 subbands
              Record frame_LL
   *********************************************************/
    split_ex (frame, V/2, H/2, ask_filter);
   /*********************************************************
               Descomposes a frame:10 subbands
              Record frame_LL
   *********************************************************/
    split_ex (frame, V/4, H/4, ask_filter);
/*********************************************************************
          Write data  from the file
 *********************************************************************/
                        frame_write(frame,nome_out,V,H );
                        for (i=0; i<V; i++){
    free(frame[i]);
                        }
  free(frame);
  printf("Fim do programa\n");
```

Parecer da Fapesp

FAPESP
FUNDAÇÃO DE AMPARO À PESQUISA DO ESTADO DE SÃO PAULO

Formulário para parecer de Acompanhamento de Assessoria Científica
Bolsa de Doutoramento no País
(Este formulário está disponível em formato eletrônico no endereço **saturno.fapesp.br**)

Processo **97 / 14325 - 4**

Interessado(a): **Vicente Idalberto Becerra Sablon**

√ RESERVA TÉCNICA
E
RELATÓRIO CIENTÍFICO

Parecer cuja cópia xerográfica será enviada ao interessado:
(Comentários, críticas e sugestões têm se mostrados úteis para o aprimoramento de projetos.)

O bolsista nesse período realizou simulações para o codificador de vídeo, com avaliações subjetivas e objetivas de resultados, contribuindo para o conhecimento da área. Considero muito relevantes os resultados. O Relatório foi bem escrito, coerente e bem estruturado.

RELATÓRIO APROVADO

Parecer da Fapesp

FAPESP
FUNDAÇÃO DE AMPARO À PESQUISA DO ESTADO DE SÃO PAULO

RECEBIDO
19 SET 2002
FAPESP

Formulário para parecer de Acompanhamento de Assessoria Científica
Bolsa de Desenvolvimento no País

(Este formulário está disponível em formato eletrônico no endereço **saturno.fapesp.br**)

Processo **97 / 14325 - 4**

Interessado(a): **Vicente Idalberto Bezerra Sablón**

V RESERVA TÉCNICA
E
RELATÓRIO CIENTÍFICO

Parecer cuja cópia xerográfica será enviada ao interessado:
(Comentários, críticas e sugestões têm se mostrados úteis para o aprimoramento de projetos.)

O bolsista fez um trabalho profundo de compreensão, análise e comparação de algoritmos de aplicação de tranformada de wavelets na codificação de vídeo, apresentando resultados quantitativos interessantes.

No próximo período, espera-se que as idéias abordadas sejam transformadas em contribuições, em formato de artigos especializados, em futuro próximo.

RELATÓRIO APROVADO

Lista de Publicações

[1] V. I. Sablón, Y Iano "Motion- Estimation using Partition, Aggregation and Condition Coding and Shift Invariant Property". *IEEE Trans. On Circuits Systems. (Submetido)*

[2] V. I. Sablón, Y Iano "A Transformada Wavelets no Processamento do Sinal de Vídeo". Curso Tutorial. Convidado. *Anais Congreso Internacional de Ingenieria Eletrônica, Elétrica.* Cusco. Peru Julho 2003.

[3] V. I. Sablón, Y Iano "Subsistema Codificador de Imagens baseado na Transformada Wavelet para a Codificação dos Sinais de Vídeo digital e HDTV". Curso Tutorial. Convidado. *Anais Congreso Internacional de Ingenieria Eletrônica, Elétrica.* Cusco. Peru Julho 2003..

[4] V. I. B. Sablón, Y. Iano, "Televisão Digital de Alta Definição-HDTV ". *Revista do Instituto Nacional de Telecomunicações.v.3 n.1 p. 1-10.*. São Paulo. Abril 2000

[5] L. R. Mendez, V. I. B. Sablón, Y. Iano, R T. Prieto. "Subsistema de Compressão e Codificação do Sinal de Video dos Padrões de HDTV – Parte II". *Revista do Instituto Nacional de Telecomunicações.v.3 n.1 p. 19-27.* São Paulo. Abril. 2000.

[6] V. I. B. Sablón, Y. Iano, R T. Prieto, L. R. Mendez. "Subsistema de Compressão e Codificação do Sinal de Video dos Padrões de HDTV – Parte I*". Revista do Instituto Nacional de Telecomunicações.v.3 n.1 p. 10-18.* São Paulo. Abril. 2000.

[7] V. I. B. Sablón, Y. Iano. "El Sistema Digital de HDTV en el Brasil". *1er Congreso Internacional en Eléctronica y Telecomunicaciones. Anais. "Inteligência 99"*. Instituto Tecnológico de Estúdios Superiores de Monterrey (ITESM). Mexico DF. Mexico. Octubre 1999.

[8] V. I. B. Sablón, Y. Iano. "Perpectivas e Tendências Atuais do Sistema Digital de HDTV no Brasil". *VI Congreso Internacional de Ingeniería Electrónica, Eléctrica y de Sistemas. INTERCON'99. Anais IEEE-UNI. Lima Perú. Agosto* 1999.

[9] V. I. B. Sablón, J. G. Chiquito, F. Araújo, A. L. Kume, "Sistema Operacional, para uma Rede Automatizada de Calibração de Instrumentos com Interfaceamento IEEE-488", 3rd International Conference on Electrical Metrology , *Anais . Rio de Janeiro. Brasil. Setembro.*1998.