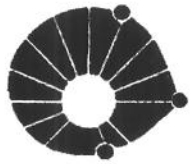
UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE COMUNICAÇÕES



UNICAMP

TESE DE DOUTORADO

PROCEDIMENTOS PARA TRATAMENTO E COMPRESSÃO DE IMAGENS E VÍDEO UTILIZANDO TECNOLOGIA FRACTAL E TRANSFORMADAS *WAVELET*.

Fernando Silvestre da Silva

Orientador: Prof. Dr. Yuzo Iano

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

Banca Examinadora:

Prof. Dr. Phillip Mark Seymour Burt (POLI-USP)

Prof. Dr. Evaldo Gonçalves Pelaes (UFPA)

Prof. Dr. Dalton Soares Arantes (FEEC - UNICAMP)

Prof. Dr. Vicente Idalberto Becerra Sablón (UNISAL - Campinas)

Campinas, SP – Brasil

Setembro– 2005

Este exemplar corresponde a redação final da tese defendida por Fernando Silvestre da Silva e aprovada pela Comissão julgada em 23 / Setembro / 2005

Yuzo Iano
Orientador

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

Si38p

Silva, Fernando Silvestre da
Procedimentos para tratamento e compressão de imagens e vídeo utilizando tecnologia fractal e transformadas *wavelet* / Fernando Silvestre da Silva. --Campinas, SP: [s.n.], 2005.

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

1. Televisão digital. 2. Compressão de imagens. 3. Wavelet (matemática). 4. Fractais. 5. Compressão de dados (Telecomunicações). 6. Processamento de imagens – Técnicas digitais. 7. Teoria da codificação. 8. Comunicações digitais. I. Iano, Yuzo. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Titulo em Inglês: Procedures for image and video compression using fractal technology and wavelet transforms
Palavras-chave em Inglês: Digital television, Image compression, Wavelet, Fractals, Data compression (telecommunication), Digital image processing, Coding theory e Digital communications
Área de concentração: Telecomunicações e Telemática
Titulação: Doutora em Engenharia Elétrica
Banca examinadora: Phillip Mark Seymour Burt, Evaldo Gonçalves Pelaes, Dalton Soares Arantes, Vicente Idalberto Becerra Sablón
Data da defesa: 23/09/2005

RESUMO

A excelente qualidade visual e taxa de compressão dos codificadores fractais de imagem tem aplicações limitadas devido ao exaustivo tempo de codificação inerente. Esta pesquisa apresenta um novo codificador híbrido de imagens que aplica a velocidade da transformada wavelet à qualidade da compressão fractal. Neste esquema, uma codificação fractal acelerada usando a classificação de domínios de Fisher é aplicada na sub-banda passa-baixas de uma imagem transformada por wavelet e uma versão modificada do SPIHT (*Set Partitioned in Hierarchical Trees*) é aplicada nos coeficientes remanescentes. Os detalhes de imagem e as características de transmissão progressiva da transformada *wavelet* são mantidas; nenhum efeito de bloco comuns às técnicas fractais é introduzido, e o problema de fidelidade de codificação comuns aos codificadores híbridos fractal-*wavelet* é resolvido. O sistema proposto reduz o tempo médio de processamento das imagens em 94% comparado com o codificador fractal tradicional e um ganho de 0 a 2,4dB em PSNR sobre o algoritmo SPIHT puro. Em ambos os casos, o novo esquema proposto apresentou melhorias em temos de qualidade subjetiva das imagens para altas, médias e baixas taxas de compressão.

ABSTRACT

The excellent visual quality and compression rate of fractal image coding have limited applications due to exhaustive inherent encoding time. This research presents a new fast and efficient image coder that applies the speed of the wavelet transform to the image quality of the fractal compression. In this scheme, a fast fractal encoding using Fisher's domain classification is applied to the lowpass subband of wavelet transformed image and a modified SPIHT coding (Set Partitioning in Hierarchical Trees), on the remaining coefficients. The image details and wavelet progressive transmission characteristics are maintained; no blocking effects from fractal techniques are introduced; and the encoding fidelity problem common in fractal-wavelet hybrid coders is solved. The proposed scheme provides an average of 94% reduction in encoding-decoding time compared to the pure accelerated Fractal coding results, and a 0-2,4dB gain in PSNR over the pure SPIHT coding. In both cases, the new scheme improves the subjective quality of pictures for high, medium and low bit rates.

OFERECIMENTO

"Aquele que habita no esconderijo do Altíssimo, à sombra do Onipotente descansará.
Direi do Senhor: *Ele é* o meu Deus, o meu refúgio, a minha fortaleza, e n'Ele confiarei.
Porque *Ele* te livrará do laço do passarinheiro, e da peste perniciosa.
Ele te cobrirá com as suas penas, e debaixo das suas asas estarás seguro: e sua verdade é escudo e broquel".

Salmos, 91:1-4

"Porque Tu, ó Senhor, és o meu refúgio! O Altíssimo é a Tua habitação.
Nenhum mal te sucederá, nem praga alguma chegará à tua tenda.
Porque aos Seus anjos dará ordens a teu respeito, para te guardarem em todos os teus caminhos.
Eles te sustentarão nas suas mãos para que não tropeces com teu pé em pedra." *Salmos, 91:9-12*

"Uma noite eu tive um sonho... Sonhei que estava andando na praia com o Senhor, e através do céu, passavam cenas da minha vida. Para cada cena que se passava, percebi que eram deixados dois pares de pegadas na areia, um era meu e outro era do Senhor.
Quando a ultima cena da minha vida passou diante de nós, olhei para traz, para as pegadas na areia, e notei que muitas vezes no caminho da vida havia apenas um par de pegadas na areia. Notei também que isto aconteceu nos momentos mais difíceis e angustiosos do meu viver. Isso aborreceu-me, então perguntei ao Senhor:
–Senhor, Tu me disseste que uma vez que resolvi te seguir, Tu andarias sempre comigo, em todo o meu caminho, mas notei que durante as maiores tribulações do meu viver, havia apenas um par de pegadas na areia. Não compreendo porque nas horas em que necessitava de Ti, tu me deixastes...
O Senhor respondeu:
–Meu precioso filho, Eu Te Amo e jamais te deixaria nas horas de tua prova e de teu sofrimento. Quando vistes na areia apenas um par de pegadas, foi exatamente aí, que Eu te carreguei nos braços."

"Pegadas na Areia", autor desconhecido

À Deus eu ofereço este trabalho, ao seu Amor infinito e à sua Divina Providência que tanto ajudam a seus filhos que caminham nesta terra.

AGRADECIMENTO À FAPESP

Agradecimentos especiais à **FAPESP – Fundação de Amparo à Pesquisa do Estado de São Paulo** – ao apoio indispensável que tem sido oferecido à este pesquisador através dos Projetos: "**Algoritmo de Medição de Efeito de Bloco em Ambiente MPEG-2 (PDI)**", Processo FAPESP no. 98/05318-7; "**Procedimentos para Medição e Minimização de Distorções Decorrentes do Processamento Digital de Imagem (PDI) – Restauração de Imagem**", Processo FAPESP no. 98/12898-0 e "**Procedimentos para Tratamento e Compressão de Imagens Utilizando Tecnologia Fractal (PDI)**", Processo FAPESP Nº 00/11157-8 que viabilizaram a realização desta pesquisa, desde a Iniciação Científica até o Doutorado, provendo suporte financeiro e possibilitando a aquisição dos recursos e equipamentos fundamentais para o desenvolvimento e conclusão desta pesquisa.

AGRADECIMENTOS

Pesquisar é a arte de tentar e errar com tal persistência que, enfim, se encontra o caminho certo. Ao contrário de muitas outras formas de arte, pesquisar requer não somente um esforço pessoal, mas uma congregação coletiva de pessoas e instituições com o intuito de atingir um objetivo. A presente pesquisa não foi diferente e, por isso, ao atingir o objetivo final, eu gostaria de agradecer à todas as pessoas que direta ou indiretamente me ajudaram a chegar nesse momento.

Primeiramente, eu gostaria de agradecer ao meu orientador, Prof. Dr. Yuzo Iano, que com sua enorme experiência e sabedoria tanto me ajudou nesse árduo caminho. Seus conselhos me deram a força, a paciência e o estímulo nas doses e momentos corretos durante estes quase 7 anos desde a Iniciação Científica. Muito obrigado por tudo.

À Universidade Estadual de Campinas – UNICAMP – por ter me proporcionado a oportunidade de estudar e concretizar o meu objetivo de concluir o doutorado. À FAPESP – Fundação de Amparo à Pesquisa do Estado de São Paulo – pelo apoio recebido nos projetos de iniciação científica, de mestrado e de doutorado.

À todos os professores da Faculdade de Engenharia Elétrica e Computação da Unicamp, que tanto contribuíram na minha formação acadêmica e profissional. Aos funcionários da FEEC, Lúcia, Noêmia, Giane, Mazé, Érgio, Nelson e tantos outros que sempre me ajudaram com extrema boa vontade e eficiência.

Aos meus pais Damásio e Lúcia, pelo amor, dedicação, apoio e por todos os sacrifícios, que somente os nossos pais são capazes de fazer, para que eu chegasse até aqui. Aos meus irmãos Sandra e Fábio que, apesar da distância, sempre estiveram prontos a conversar e me apoiar nos momentos difíceis.

Aos meus sogros, Sr. Ary e Dona Teresa que estiveram sempre presentes ajudando à superar nossas dificuldades e que tantas alegrias nos trouxeram nesses anos e que ainda trarão nos próximos.

Aos meus amigos, Rodrigo, Darclê e Emílio que proporcionaram incontáveis e divertidas conversas, e pela paciência com o meu excessivo "papo técnico".

Aos meus amigos Vicente e Zaida que, através de seus conselhos, palavras positivas e seu próprio exemplo, muito colaboraram para conseguirmos galgar este degrau do doutorado.

À minha esposa Ana Lúcia, uma mulher brilhante em todos os sentidos, sem a qual eu não sonharia chegar a este momento e que, mais do que qualquer outra pessoa no mundo compreendeu e ajudou a corrigir meu erros e se alegrou com meus acertos.

À Deus, pela Divina Providência.

Fernando Silvestre da Silva

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos Principais

$\omega_i(x)$ – Transformação afim contrativa

$W(x)$ – Conjunto de Transformações contrativas

$\mathfrak{R}$ – Conjunto dos Números Reais

$\mathfrak{R}^2$ – Plano Euclidiano

$\mathcal{H}(X)$ – Espaço dos subconjuntos compactos e não vazios de X

$\lambda(x)$ – Operador de Projeção de Saupe

$\psi(x)$ – *Wavelet*-mãe ou *wavelet*-básica

$\phi(x)$ – Função-escala

$\psi_{j,n}(x)$ – *Wavelets*- filhotes, versões deslocadas e dilatadas ou comprimidas da *wavelet*-básica.

$\phi_{j,n}(x)$ – Versões deslocadas e dilatadas ou comprimidas da função-escala.

V_{2^j} – Espaço das funções quadraticamente integráveis com resolução 2^j

W_{2^j} – Espaço das funções com resolução 2^j ortogonal à V_{2^j} em relação à $V_{2^{j+1}}$.

Abreviaturas

2D – Bidimensional

AC – *Alternate Current*

BPP – *Bits* por *Pixel*

DC – *Direct Current*

EZW – *Embedded Zerotree Wavelet Coding*

FIR – *Finite Impulse Response*

GOP – *Group of Pictures*

IEEE – *Institute of Electrical and Electronics Engineers*

IFS – *Iterated Function System*

IIR – *Infinite Impulse Response*

JPEG – *Joint Pictures Expert Group*

LIP – *List of Insignificant Pixels*

LIS – *List of Insignificant Sets*

LSP – *List of Significant Pixels*

MPEG – *Moving Pictures Expert Group*

MRME – *Multi Resolution Motion Estimation*

PB – Passa Baixas

PIFS – *Partitioned Iterated Function System*

Pixel – *Picture Element*

PSNR – *Peak Signal to Noise Ratio*

QPIFS – *Quadtree Partitioned Iterated Function System*

RMS – *Root Mean Square*

SF – Série de Fourier

SPIHT – *Set Partitioning in Hierarchical Trees*

SW – Série da *Wavelet*

TF – Transformada de Fourier

TW – Transformada *Wavelet*

TWC – Tranformada *Wavelet* Contínua

TWD – Tranformada *Wavelet* Discreta

Capítulo 1
Introdução

1.1 - Imagens e vídeo digitais

Nos anos recentes, os avanços da tecnologia eletrônica de consumo e de comunicações possibilitaram que as imagens e vídeo digitais ocupassem uma parcela cada vez maior do mundo da informação. A queda nos preços de aparelhos de *scanner*, câmaras fotográficas e filmadoras digitais, impressoras gráficas e *webcams* trouxe ao consumidor médio a possibilidade de integrar esta nova forma de armazenamento à vida cotidiana.

Nesse âmbito, os algoritmos de compressão digital de imagens vêm se tornando cada vez mais importantes para o dia-a-dia. Junto com esta importância crescem também os investimentos e as pesquisas para o melhoramento das tecnologias existentes e para a procura de novas técnicas que ofereçam melhores desempenhos ou alternativas para os algoritmos tradicionais.

Das novas tecnologias que estão sendo desenvolvidas nos últimos anos, duas ganham destaque cada vez maior na bibliografia técnica especializada: as transformadas *wavelet* discretas (TWD) e, mais recentemente, a codificação fractal de imagens.

Como características principais, as TWD apresentam a possibilidade de transmissão hierárquica, algoritmos rápidos e boa capacidade de concentrar a energia do sinal em poucos coeficientes, além de não introduzir artefatos de bloco. Essas vantajosas características permitiram a inserção da TWD em importantes padrões recentes de compressão de imagem e vídeo, tais como o JPEG2000, MPEG-4 e MPEG-7.

A compressão fractal, por sua vez, apresenta também suas vantagens tais como a rápida decodificação, boa compressão e o fato de representar uma nova perspectiva sobre o problema da codificação digital de imagens. Por se tratar de uma técnica ainda pouco explorada e apresentar um alto potencial para compressão, a compressão fractal se apresenta como um vasto campo para a pesquisa. Um dos aspectos que encorajam a codificação fractal é o fato de este tipo de codificação ser completamente paralelizável, permitindo, assim, que um hardware paralelo especializado consiga reduzir significativamente o tempo de codificação.

Estas tecnologias, no entanto, não podem ser vistas apenas como substitutas das técnicas existentes, mas sim como aliadas para o desenvolvimento de algoritmos híbridos que objetivem

combinar as vantagens individuais, para atingir um desempenho melhorado em termos de taxa-distorção. O presente trabalho trata exatamente de uma técnica que combina fractal e *wavelet* que pretende apresentar melhorias não somente de taxa-distorção, como também tempo de processamento e qualidade subjetiva (redução de artefatos de compressão).

1.2 - Contribuições e Organização do Trabalho

O objetivo deste trabalho é propor dois esquemas híbridos, sendo o primeiro um algoritmo de compressão de imagens baseado em transformada *wavelet* e codificação fractal de imagens que apresente um desempenho melhorado com relação à qualidade subjetiva, tempo de processamento, PSNR e taxa de compressão de forma a possibilitar que, através da incorporação de técnicas já conhecidas de compressão temporal, seja desenvolvido um segundo algoritmo na área de codificação digital de vídeo.

Através destes esquemas, nos diversos experimentos realizados para imagens obteve-se uma melhoria da qualidade visual e PSNR, em conjunto com um aumento na velocidade de processamento de 94% em relação à codificação fractal pura, tornando viável a exploração desse algoritmo para a codificação de vídeo. O codificador de vídeo também obteve resultados muito promissores apresentando um PSNR médio semelhante ao algoritmo do MPEG-2 MP@ML para seqüências de *720x480@30fps* codificadas a *2* Mbps.

Pode-se então resumir as principais contribuições desse trabalho como:

a) Aprofundar o estudo de teoria de fractais e transformada *wavelet* discreta de forma a tentar explorar as características comuns entre estas técnicas com a intenção de gerar algoritmos eficientes de codificação no ambiente de processamento digital de sinais, imagens e vídeo;

b) A implementação de um algoritmo de codificação de imagens híbrido Fractal-Wavelet, utilizando a excelente recuperação de detalhes da codificação fractal de imagens para obter resultados melhorados de codificação em relação aos algoritmos clássicos da codificação em transformada *wavelet* pura através da aplicação do QPIFS (*Quadtree Partitioned Iterated Function System*) à sub-imagem passa-baixas da imagem original decomposta por TWD de forma a melhorar sua codificação deixando uma maior parcela de bits para as sub-imagens subseqüentes que serão codificadas por SPIHT (*Set Partitioned in Hierarchical Trees*);

c) O desenvolvimento de um esquema de codificação de vídeo que incorpora algoritmos de estimação e compensação de movimento em domínio *wavelet* (MRME - *Multi-Resolution Motion Estimation*) ao sistema de codificação de imagens de forma a possibilitar que apresente melhor qualidade de vídeo recuperado em relação ao sistema MPEG-2.

A apresentação do restante desse trabalho foi estruturada da seguinte forma:

PARTE I: Fractais

Capítulo 2: Teoria de Fractais

Nesse capítulo são apresentados os conceitos básicos de fractais, seu princípio de geração e suas bases matemáticas incluindo uma revisão dos conceitos de topologia que são necessários para a compreensão da definição de fractais;

Capítulo 3: Construção de Fractais: Sistema de Funções Iterativas – IFS

Esse capítulo apresenta o método de geração de fractais através de funções iterativas denominado IFS - *Iterated Function System* - desenvolvido por Hutchinson [1] e que foi utilizado por Barnsley [2] como base para a aplicação da teoria de fractais no problema da codificação de imagens digitais. Também são apresentados o PIFS – sistema particionado de funções iterativas – desenvolvido por Jacquin [3] e o conceito de auto-similaridade por partes que formam a base de todos os codificadores fractais de imagens encontrados na literatura atual.

Capítulo 4: Codificadores Fractais de Imagem

Apresentam-se neste capítulo alguns algoritmos de codificação de imagens utilizando tecnologia fractal incluindo os sistemas que foram utilizados como base de comparação para o algoritmo híbrido desenvolvido durante essa pesquisa. São abordados também o conceito de *quadtrees* e alguns métodos de classificação de domínios utilizados para acelerar a codificação fractal pura PIFS.

PARTE II: Transformadas *Wavelet*

Capítulo 5: Transformada *Wavelet* Discreta

Nesse capítulo é apresentada uma introdução sobre a transformada *wavelet* continua (TWC) uni e bidimensionais, pelas séries *wavelet* (SW) e transformada *wavelet* discreta (TWD). São também apresentados os conceitos de codificação por sub-banda e o algoritmo rápido de Mallat para o cálculo da TWD.

Capítulo 6: Análise Multi-resolução

Nesse capítulo apresenta-se a base matemática da decomposição e da síntese através da TWD para os casos ortogonal uni e bi-dimensional, bem como as definições dos conceitos de sub-bandas de aproximação e detalhamento comumente utilizadas em *wavelet*. Também se analisa o

efeito da ortogonalidade e suas restrições discutindo-se a necessidade do relaxamento dessas restrições para se atingir *wavelets* biortogonais, que apresentam características mais propícias para a aplicação em imagens.

Capítulo 7: Biortogonalidade

Neste capítulo são abordadas a decomposição e síntese da TWD para o caso biortogonal 1D e 2D, que apresenta desempenho excelente para a compressão de imagens devido às suas características: suporte compacto, simetria e bom compromisso entre regularidade e comprimento do filtro. A importância dessas características serão melhor abordadas no Capítulo 9.

Capítulo 8: Regularidade, Seleção das *Wavelets* e SPIHT

Ao contrário das transformadas tradicionais, a transformada *wavelet* não possui apenas uma base possível. Na verdade, a teoria de *wavelets* define um conjunto praticamente infinito de famílias de funções-base que podem ser utilizadas. Assim, torna-se necessário o estudo das características fundamentais de cada tipo de *wavelet* de forma a possibilitar uma escolha do conjunto de funções-base mais adequado para cada caso. Nesse capítulo são descritas essas características e destacando as que são mais importantes para o caso da compressão de imagem e vídeo e justificando a escolha da *wavelet spline* biortogonal 9-7 que foi utilizada em nosso trabalho.

PARTE III: Codificação de Vídeo.

Capítulo 9: Conceitos Básicos de Codificação de Vídeo

Nesse capítulo serão abordados conceitos básicos de vídeo, iniciando com o conceito de frame e de seqüência de vídeo, passando por GOP (*Group of Pictures*), compressão temporal e estimação/compensação de movimento. A intenção é introduzir a terminologia da compressão de vídeo digital possibilitando uma melhor compreensão do algoritmo de codificação de vídeo desenvolvido nesse trabalho.

Capítulo 10: Algoritmos de Estimação de Movimento em Domínio *Wavelet*

Nesse capítulo será apresentado o algoritmo de compressão temporal no domínio *wavelet* denominado estimação de movimento multi-resolução (MRME) desenvolvido por Zhang e Zafar [4]. Também serão apresentadas as variações desse algoritmo propostas por Swammy *et. al.* [5], sendo que uma delas será a utilizada para o codificador de vídeo proposto.

PARTE IV: Algoritmos Propostos, Resultados e Conclusões

Capítulo 11: Codificadores Propostos

Neste capítulo serão apresentados os codificadores de imagem e de vídeo desenvolvidos durante esse trabalho incluindo detalhes de implementação e princípios de funcionamento. A idéia central para o codificador de imagens foi utilizar a codificação fractal na sub-imagem de aproximação de forma a obter uma melhor codificação dessa sub-imagem e permitindo que as camadas posteriores pudessem receber mais bits, obtendo, assim, uma melhor qualidade de imagem e um baixo tempo de processamento. E a partir desse algoritmo, incorporando técnicas de estimação temporal, desenvolver o algoritmo de codificação híbrido fractal-*wavelet* de vídeo.

Capítulo 12: Apresentação e Análise dos Resultados

Serão apresentados aqui os resultados experimentais obtidos visando uma análise da qualidade visual, PSNR, taxa de compressão e tempo de processamento para todos os algoritmos de codificação de imagens testados. Também são apresentados e analisados os resultados do codificador de vídeo em comparação com o MPEG-2 TM5.

Capítulo 13: Conclusões e Sugestões para Trabalhos Futuros

À luz dos resultados obtidos e de sua análise, nesse capítulo apresentam-se as conclusões que puderam ser extraídas e as perspectivas de melhorias e desenvolvimentos futuros que podem colaborar para a ampliação e refinamento desse trabalho. Espera-se que esse trabalho tenha colaborado para expor o alto potencial tanto dos sistemas de compressão fractal puro como em conjunto com as transformadas *wavelet* na aplicação de codificação de sinais, imagem e vídeo digitais.

Capítulo 2
Teoria de Fractais

2.1 - Introdução

Dentre os padrões atuais de compressão digital de imagem e vídeo, a grande maioria utiliza a técnica de codificação por transformada. Esta técnica consiste na aplicação de uma transformação na imagem original que permita a eliminação/redução dos componentes em alta freqüência do sinal, dando prioridade ao armazenamento dos componentes da baixa freqüência. Estes sistemas têm sua maior vantagem na complexidade computacional relativamente baixa que permite que suas implementações atinjam codificações em tempo real preservando uma boa qualidade.

No entanto, o aumento exponencial da capacidade de processamento dos equipamentos nos últimos anos tem estimulado o aumento das pesquisas em sistemas alternativos de compressão que utilizem princípios novos que busquem a melhoria da qualidade dos dados codificados. Dentre estes esquemas, um dos maiores destaques é a Codificação Fractal de Imagens, cuja base matemática foi inicialmente proposta em 1989 por M. Barnsley [2] e que tem despertado interesse de diversos pesquisadores no mundo inteiro.

O modelo fractal é muito diferente dos modelos tradicionais. Até hoje, quase todos os modelos matemáticos aplicados em engenharia encorajam aproximações suaves, contínuas e diferenciáveis. Isso implica na hipótese de "retificabilidade", ou seja, qualquer parte do modelo que sofra ampliação é assumida por hipótese como sendo plana e suave. Essa hipótese é muito simples mas, apesar de gerar bons resultados para um grande número de aplicações práticas, é bastante restritiva num âmbito mais geral.

Assim, ampliando-se a foto de uma floresta, via modelo clássico, ver-se-á apenas uma grande área verde plana e uniforme. É nesse ponto que se apresenta a grande diferença da aproximação fractal, na qual essa ampliação criaria uma série de detalhes baseados na informação da imagem original (não-ampliada). Logo, ao invés de supor grau de complexidade zero para as ampliações, o modelo fractal supõe a existência de uma "Auto-Similaridade" entre as escalas, ou seja, que a região ampliada tem o mesmo grau de complexidade apresentado na imagem original se assemelhando mais fielmente ao mundo real.

Comparando o princípio de modelamento fractal com os sistemas tradicionais, pode-se notar claramente a diferença entre os conceitos. Segundo Ning Lu [6], "Considerando os modelos clássicos como uma aproximação de ordem zero, o modelamento fractal pode ser visto como uma aproximação de primeira ordem".

2.2 - Princípio Fractal

Considere-se inicialmente uma folha branca de papel com um sistema de coordenadas (x,y) marcado nela e no qual escolhe-se um ponto arbitrário A. Selecionando aleatoriamente uma das transformações afins listadas abaixo e aplicando-a neste ponto, obtém-se as coordenadas de um novo ponto B [6]. Marque-se o ponto B no papel.

$$\begin{aligned}
&\omega_1 : (x, y) \rightarrow (0.85x+0.04y,\ -0.04x+0.85y+40),\\
&\omega_2 : (x, y) \rightarrow (0.20x-0.26y,\ 0.23x+0.22y+40),\\
&\omega_3 : (x, y) \rightarrow (-0.15x+0.28y,\ 0.26x+0.24y+11),\\
&\omega_4 : (x, y) \rightarrow (0,\ 0.16y).
\end{aligned} \tag{2.1}$$

Seleciona-se novamente uma transformação e aplica-se no ponto B gerando um ponto C. Marca-se este ponto. Repetindo esse processo indefinidamente e sendo pacientes e persistentes o suficiente, uma folha de samambaia surgirá no papel (Fig. 2.1a).

Define-se fractal como sendo uma imagem ou figura que pode ser completamente descrita com suas texturas infinitamente detalhadas por um algoritmo matemático. Na maioria dos casos, essas texturas e detalhes não podem ser preditos pela geometria clássica.

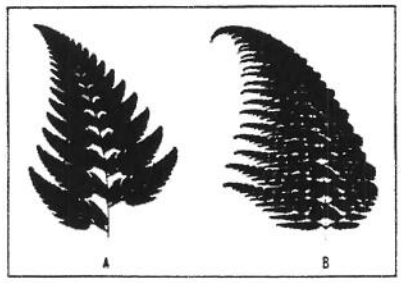


Fig. 2.1 – Folhas de samambaia: a) 2D; b) 3D. [7]

No caso, a folha de samambaia é um exemplo de fractal que pode ser definido por quatro transformações de *6* coeficientes cada, necessitando portanto de somente *24* números: *85, 4, 0, -4, 85, 40; 20, -26, 0, 23, 22, 40; -15, 28, 0, 26, 24, 11; 0,0,0,0,16,0.*

Essas quatro transformações afins formam um Sistema de Funções Iterativas (*Iterated Function System*, IFS). O fractal imagem da folha de samambaia é matematicamente chamado de *atrator* desse IFS. Cada repetição do processo é chamada de iteração [6].

Transformações afins planares podem ser generalizadas para o espaço 3D. Nesse caso, cada transformação deve ser caracterizada não por *6*, mas por *12* coeficientes. No caso da folha de samambaia apresentada na Fig. 2.1b [6]:

85, 0, 0, 0, 0, 85, 13, 81, 0, -9, 85, 0;

31, -34, 0, 0, 12, 21, 0, 24, 0, 0, 30, 0;

-29 ,33, 0, 0, 12, 19, 0, 66, 0, 0, 30, 0;

0, 0, 0, 0, 0, 20, 0, 0, 0, 0, 0, 0.

Uma analogia simples foi feita por Fisher [7] para explicar esses princípios básicos de funcionamento da idéia fractal. Essa analogia consiste num tipo especial de fotocopiadora que reduz a imagem a ser copiada pela metade e a reproduz três vezes na cópia de saída, como mostrado na Fig. 2.2.

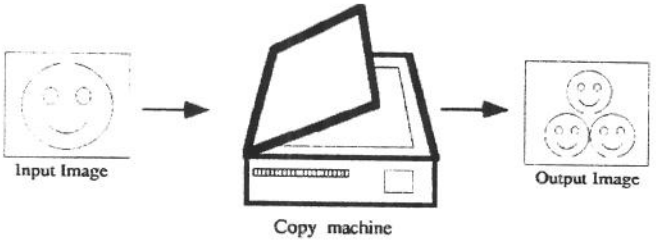


Fig.2.2 – Exemplo de Fotocopiadora Especial. [7]

A idéia básica consiste em passar essa cópia de saída pela copiadora, num processo recursivo. Após várias iterações desse processo, percebe-se na Fig. 2.3 que mesmo com diferentes imagens de entrada, todas as cópias de saída parecem convergir para a mesma imagem final, mostrada na Fig. 2.3c. Essa imagem é o atrator para essa máquina fotocopiadora.

Uma vez que a fotocopiadora reduz a imagem de entrada, qualquer imagem inicial será reduzida a um ponto à medida que aumentam as iterações. Logo, a imagem inicial não afeta o atrator final; de fato somente a posição e orientação das cópias (transformações) determinam como a imagem final se parecerá [7].

Diferentes transformações levam a diferentes atratores e possuem a limitação de terem de ser "contrativas", ou seja, a distância entre dois pontos quaisquer da imagem de saída (após transformação), precisa ser menor que a distância entre os pontos correspondentes na imagem de entrada [7]. A distância entre dois pontos P_1 e P_2 é dada por:

$$d(P_1,P_2)=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2} \tag{2.2}$$

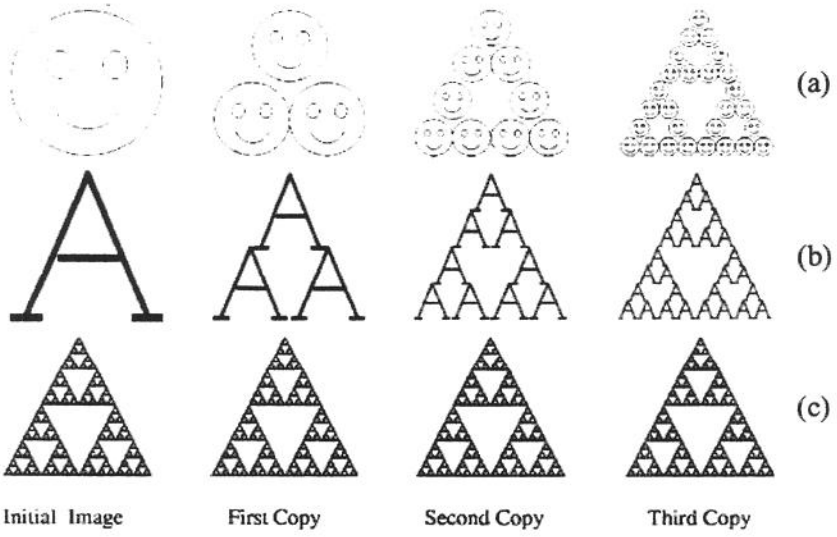


Fig. 2.3 – Primeiras 3 cópias geradas pela fotocopiadora para 3 imagens diferentes.

Na prática, transformações na forma (2.3) permitem gerar uma grande variedade de interessante atratores. Estas transformações podem torcer, deslizar, rotacionar, escalar ou deslocar a imagem de entrada, dependendo dos valores dos coeficientes.

$$\omega_i\begin{bmatrix}x\\y\end{bmatrix}=\begin{bmatrix}a_i & b_i\\c_i & d_i\end{bmatrix}\cdot\begin{bmatrix}x\\y\end{bmatrix}+\begin{bmatrix}e_i\\f_i\end{bmatrix} \tag{2.3}$$

A Fig. 2.4 mostra alguns exemplos de transformações com seus respectivos atratores e a ampliação de uma região de cada atrator. As transformações são aplicadas sobre uma imagem inicial em forma de "L" (lembremo-nos que o resultado independe dela). Essa forma ajuda a visualizar as transformações [7].

O primeiro exemplo da Fig. 2.4 mostra as transformações usadas na fotocopiadora da Fig. 2.2. O segundo exemplo é muito similar ao primeiro, entretanto invertendo uma das cópias, gerando um atrator diferente. O último exemplo é a folha da samambaia da Fig. 2.1. Os dois primeiros exemplos da Fig. 2.4 são constituídos de três transformações, enquanto o último, como se vê, utiliza quatro. Note que realmente mudanças nas transformações, ainda que pequenas, geram diferentes atratores.

A característica comum a todos os atratores frutos deste tipo de transformação é que, ao ampliarmos uma de suas regiões com detalhes, ver-se-á uma cópia da imagem original, ou seja,

cada atrator é formado por cópias reduzidas e transformadas de si mesmo, possuindo assim detalhes em todas as escalas. Essa característica dos fractais é chamada "Auto-Similaridade" e é de grande importância para os esquemas de compressão Fractal como será visto no próximo capítulo. Este método de gerar fractais foi criado por John Hutchinson [1].

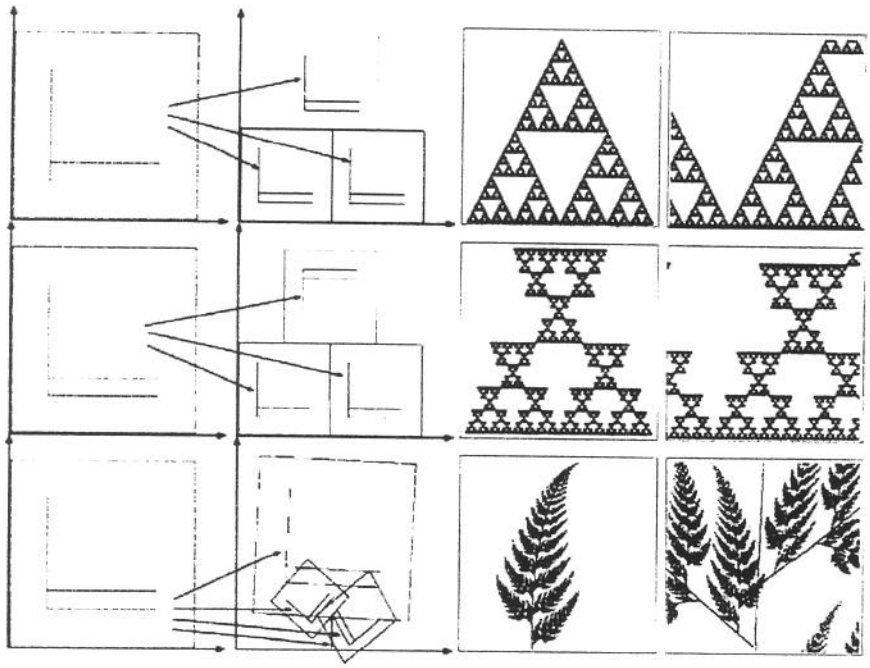


Fig. 2.4 – Transformações Afins, Atratores e Ampliação do Atrator.

Computadores de hoje não somente permitem a visualização de fractais como objetos matemáticos surpreendentes, mas também possibilitam novas formas de modelar o mundo real. A Fig. 2.5 mostra um exemplo de paisagem criada usando técnicas fractais.

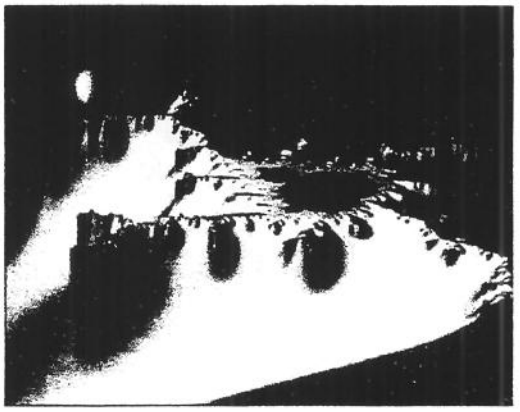


Fig. 2.5 – Paisagem composta por fractais feita por Benoit Mandelbrot.

Uma vez conseguindo produzir fractais que imitam imagens reais, Barnsley e alguns pesquisadores, propuseram uma outra questão: "Qualquer imagem real poderia ser descrita por fractais?". A busca por esta resposta levou os pesquisadores a identificarem o tipo de auto-similaridade encontrada nas imagens naturais, criando um novo campo científico: a representação fractal de imagens.

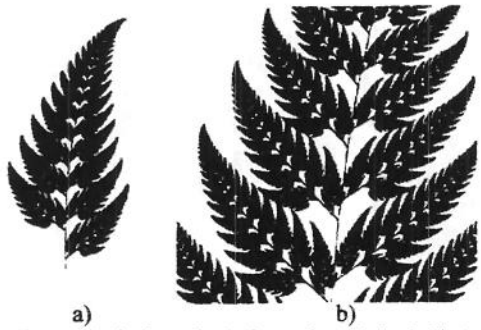
Barnsley [8] sugeriu que armazenando os coeficientes das transformações afins ao invés de armazenar a imagem como uma coleção de *pixels*, poderia gerar significativa compressão. Por exemplo, no caso da imagem folha de samambaia Fig. 2.1 (256x256, 1bpp) seriam necessários 65.536 bits. Com a tecnologia fractal só se necessitaria armazenar 24 coeficientes, que com precisão de 32 bits, leva a 768 bits armazenados. Uma vez armazenados os coeficientes, para reconstruir a imagem basta aplicar recursivamente as transformações por eles determinadas em uma imagem qualquer.

A questão primordial da compressão torna-se, então, a determinação desses coeficientes e o desenvolvimento de algoritmos rápidos para a realização desse processo.

A partir deste ponto, para se analisar melhor o processo de compressão fractal faz-se necessário a apresentação da base matemática do processo através da Teoria de Fractais.

2.3 - Conceito de Fractais

O que é um fractal? Segundo Fisher [7], os fractais não possuem uma definição formal aceita universalmente, no entanto há algumas propriedades gerais seguidas por eles. Ao olhar para um fractal como o da Fig. 2.6 [8], considerando que cada ponto preto na imagem seja a representação de um ponto em $\mathcal{R}^2$, pode-se dizer que, basicamente, um fractal é um subconjunto de pontos do espaço. Essa definição é correta, mas é incompleta e há aspectos que devem ser mais bem trabalhados de forma a torná-la mais exata e abrangente.



a) b)

Fig. 2.6 - Exemplo de fractal: a) Tamanho original; b) Ampliação.

Um dos modos citados por Fisher [7] para se definir um fractal é: "Um fractal é um subconjunto compacto de um espaço métrico completo". Como esta definição envolve alguns conceitos de Topologia de Espaços Métricos, vamos apresentar uma breve introdução destes conceitos.

2.4 - Espaço: Definição e exemplos

O primeiro aspecto que pode ser abordado é o conceito de "espaço" [8].

Definição 2.1: "Qualquer espaço X é um conjunto. Os *pontos* do espaço são os elementos do conjunto".

Observe que a notação *ponto* (em itálico) se refere ao elemento do espaço e não ao conceito de ponto euclidiano tradicional (representado como na Fig. 2.7 [8]).

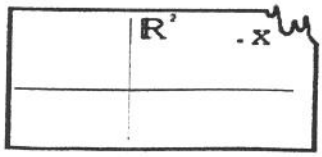


Fig. 2.7 - Um ponto em um plano euclidiano $\mathfrak{R}^2$.

Alguns exemplos de espaços são [8] [9]:

1. $X = \mathfrak{R}$, onde $\mathfrak{R}$ é o conjunto de números reais. Cada *ponto* $x \in X$ é um número real e pode ser representado por um ponto em um eixo.
2. $X = \{f \mid f: [0;1] \rightarrow \mathfrak{R}\}$. Cada *ponto* $f \in X$ é uma função real sobre o intervalo $\{x \in \mathfrak{R} \mid 0 \leq x \leq 1\}$. Note que cada *ponto* f não é um ponto no eixo x, mas sim uma função inteira que pode ser representado por um gráfico como o mostrado na Fig. 2.8.

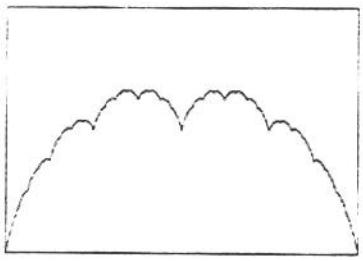


Fig. 2.8 - Exemplo de *ponto* no espaço $\{f \mid f: [0;1] \rightarrow \mathfrak{R}\}$.

3. $X = \mathfrak{R}^2$, que é o plano Euclidiano. Cada *ponto* $x \in X$ é um par ordenado (x_1,x_2) tal que $x_1,x_2 \in \mathfrak{R}$ e pode ser representado por um ponto no plano como o mostrado na Fig. 2.7.

É interessante notar que o plano $\mathfrak{R}^2$ também pode ser visto como o Produto Cartesiano do espaço $\mathfrak{R}$ com ele mesmo, ou seja, $\mathfrak{R}^2 = \mathfrak{R}\times\mathfrak{R}$. Genericamente, o Produto Cartesiano $\boldsymbol{Z}$ de dois espaços quaisquer $\boldsymbol{X}$ e $\boldsymbol{Y}$, indicado por $\boldsymbol{Z}=\boldsymbol{X}\times\boldsymbol{Y}$, é o conjunto de pares ordenados $\boldsymbol{Z}=\{z=(x,y) \mid x\in\boldsymbol{X}, y\in\boldsymbol{Y}\}$.

4. $\boldsymbol{X} = \{A \mid A\subset\mathfrak{R}^2\}$ é o espaço de todos os possíveis subconjuntos do plano euclidiano. Assim, cada *ponto* $A\in\boldsymbol{X}$ é um conjunto de pontos do plano $\mathfrak{R}^2$ e pode ser representado por um conjunto de pontos pretos em um fundo branco. Note que o fractal apresentado na Fig. 2.6 é um dos elementos deste conjunto, assim como todas as imagens pretas com fundo branco (que não tem tons de cinza e nem cores).
5. $\boldsymbol{X} = \boldsymbol{C}$ que é o plano complexo, onde qualquer *ponto* é representado por: $x = x_1 + i.x_2$, onde x_1 e x_2 são números reais e $i = \sqrt{-1}$. O plano complexo é essencialmente igual ao plano euclidiano, com exceção de que no plano complexo é definida a multiplicação entre dois pontos em $\boldsymbol{C}$ obtendo um novo ponto em $\boldsymbol{C}$:

$$x.y = (x_1 + i.x_2)(y_1 + i.y_2) = (x_1.y_1 - x_2.y_2) + i.(x_2.y_1 + x_1.y_2). \tag{2.4}$$

É interessante notar que a definição de espaço é muito geral e engloba uma ampla gama de variações e propriedades específicas. No entanto, como o interesse primário deste trabalho é o estudo de fractais, deter-se-á somente nos aspectos de maior relevância para o objetivo da tese.

Assim, dado um espaço $\boldsymbol{X}$ qualquer, é possível definir um fractal dentro deste espaço? A resposta é, em geral, negativa. Isto ocorre porque, como será visto posteriormente, os fractais existem somente em espaços chamados *espaços métricos completos*. A primeira etapa para a definição destes espaços é a definição de métrica e espaços métricos.

2.5 - Métricas e Espaços Métricos

Os espaços métricos são definidos matematicamente pelo par $(\boldsymbol{X},d)$ dados por [6] [8] [10]:

Definição 2.1: "Dado um espaço $\boldsymbol{X}$ qualquer e uma aplicação (função) d: $\boldsymbol{X}\times\boldsymbol{X} \rightarrow \mathfrak{R}$, o par $(\boldsymbol{X},d)$ é um espaço métrico se:

1. $0 < d(x,y) < \infty \Leftrightarrow x \neq y,\ \forall x, y \in \boldsymbol{X}$;
2. $d(x,x)=0\ \forall x\in\boldsymbol{X}$; (2.5)
3. $d(x,y)= d(y,x),\ \forall x, y \in \boldsymbol{X}$;
4. $d(x,z)\leq d(x,y)+ d(y,z),\ \forall x, y, z \in \boldsymbol{X}$ (Desigualdade triangular);

Se a função d obedecer a estas propriedades, então ela é chamada de métrica e a distância entre dois pontos x e y quaisquer do espaço $(\boldsymbol{X},d)$ é dada por $d(x,y)$".

Um conjunto de métricas muito utilizado é o conjunto de métricas L_p:

$$d_{L_p}(x,y) = \left(\sum_{i=1}^{n} \left| x_i - y_i \right|^p \right)^{1/p}, \text{ para } x,y \in \Re^n \tag{2.6}$$

$$d_{L_p}(f,g) = \left\| f(x),g(x) \right\|_p = \left(\int_X \left| f(x) - g(x) \right|^p dx \right)^{1/p}, \text{ para } x,y \text{ no espaço contínuo } X. \tag{2.7}$$

Dentro dessa família de métricas são incluídas a métrica do módulo ($p=1$), a métrica euclidiana ou *rms* ($p=2$) e a métrica do máximo ($p \to \infty$). A métrica *rms* é a métrica mais comumente usada no processamento digital de imagens.

2.6 - Seqüências de Cauchy e Espaços Métricos Completos

Uma vez definido o conceito de Espaços Métricos, o próximo passo é a definição da Completitude destes espaços. A definição desta propriedade é feita com o auxílio das seqüências de Cauchy apresentadas a seguir [9][11].

Definição 2.2: "Uma seqüência $\{x_n\}_{0<n<\infty}$ é considerada uma seqüência de Cauchy em um espaço métrico (X,d) se $\{x_n\} \in X$ e, para qualquer $\varepsilon \in \Re > 0$, há um número inteiro $N > 0$ tal que $d(x_n,x_m) < \varepsilon$, $\forall m,n > N$".

Ou seja, à medida que n aumenta, os pontos da seqüência $\{x_n\}$ se aproximam cada vez mais um do outro. Note que não se pode inferir que esta seqüência converge para um ponto, pois este ponto pode não existir dentro do espaço X. Um exemplo de uma seqüência de Cauchy que não converge para um ponto é: dado o espaço $X = (0;2) = \{x \mid 0<x<2,\ x \in \Re\}$ e a seqüência de Cauchy $\{x_n = 1/n\}_{1<n<\infty} \subset X$, pode-se facilmente observar que:

$$\lim_{n\to\infty} \left(\frac{1}{n} \right) = 0 \notin X. \tag{2.8}$$

Assim, pode-se definir que uma seqüência convergente em X é tal que, existe um ponto $x \in X$ tal que, para qualquer $\varepsilon \in \Re > 0$, há um número inteiro N tal que $d(x,x_n) < \varepsilon$, $\forall n > N$. E, nesse caso, o ponto x é chamado limite da seqüência e pode ser matematicamente definido por:

$$x = \lim_{n\to\infty} x_n. \tag{2.9}$$

É relativamente fácil de se provar usando a propriedade da desigualdade triangular que toda seqüência convergente é uma seqüência de Cauchy [8]. A partir disso, pode-se, então, definir o espaço métrico completo.

Definição 2.3: "Um espaço métrico X é completo se, e somente se, todas as seqüências de Cauchy $\{x_n\}_{0<n<\infty}$ em X convergem para um ponto limite $x \in X$".

Dessa forma, dada uma seqüência de Cauchy qualquer em um espaço métrico completo, então pode-se afirmar que esta seqüência converge para um ponto limite x pertencente a este espaço métrico de forma que a bola aberta $B(x,\varepsilon)$, $\forall \varepsilon \in \Re > 0$ contém infinitos pontos da seqüência.

Como um fractal é um subconjunto de um espaço métrico, o próximo passo rumo à definição de fractais é o estudo de algumas características desses subconjuntos [8][12].

Subconjuntos de Espaços Métricos

Definição 2.4: "Seja S um subconjunto do espaço métrico X (denotado por $S \subset X$), então S é aberto se, e somente se, para todo ponto $x \in S$, existe um $\varepsilon \in \Re > 0$ tal que a bola aberta $B(x,\varepsilon) \subset S$".

Por exemplo o intervalo $S=\{x \in \Re \mid 0 < x < 1\} \subset \Re$ é aberto e o intervalo $S'=\{x \in \Re \mid 0 \leq x < 1\} \subset \Re$ não é aberto. Note que se um intervalo dentro da reta real contiver qualquer uma das suas extremidades ele não será aberto. No caso de espaços multidimensionais, para que o conjunto seja aberto ele não deve conter nenhum de seus pontos limite. A definição formal de ponto limite é:

Definição 2.5: "Dado o subconjunto de um espaço métrico $S \subset X$, um ponto $x \in S$ é chamado de *ponto limite* de S se existir um seqüência convergente $\{x_n\}_{0<n<\infty} \subset S$ tal que os pontos $x_n \in \{y \in S \mid y \neq x\}$ e o seu ponto limite seja $x = \lim_{n \to \infty} x_n$".

Pode-se então definir um conjunto fechado como sendo:

Definição 2.6: "O subconjunto de um espaço métrico $S \subset X$ é fechado se, e somente se, ele contiver todos os seus pontos limite".

É interessante notar que se um conjunto qualquer contiver apenas uma parte de seus pontos limite, então ele não será aberto nem fechado e, se o conjunto for vazio ou igual ao espaço todo ele será aberto e fechado.

Pode-se agora definir outras características como conjuntos compactos e limitados:

Definição 2.7: "O subconjunto de um espaço métrico $S \subset X$ é compacto se cada seqüência infinita $\{x_n\}_{0<n<\infty}$ em S contiver uma subseqüência que tenha limite em S".

Definição 2.8: "O subconjunto de um espaço métrico $S \subset X$ é limitado se, e somente se, existir um ponto $x \in X$ e um número $r \in \Re > 0$ tal que:

$$d(x,a) < r, \quad \forall a \in S \Leftrightarrow S \subset B(x,r) \tag{2.10}$$

e é totalmente limitado se, para cada $\varepsilon \in \Re > 0$ há um conjunto finito de pontos $\{y_1, y_2, ..., y_n\} \subset S$ tal que para qualquer $x \in X$, $d(x,y_i) < \varepsilon$ para algum $y_i \in \{y_1, y_2, ..., y_n\}$".

A partir destas definições também podemos enunciar o Teorema [8]:

Teorema 2.1: "Seja (X,d) um espaço métrico completo e $S \subset X$ um subconjunto de X. Então S é compacto se, e somente se, S for fechado e totalmente limitado".

Foram apresentadas aqui as características básicas dos subconjuntos de espaços métricos mais importantes para o estudo de fractais. Abordagens mais completas a respeito de Teoria de Conjuntos e Topologia podem ser encontradas em [9] e [12]. O próximo passo agora é definir o espaço métrico onde os fractais existem: o espaço métrico $(\mathcal{H}(X),h)$.

2.7 - O Espaço Métrico $(\mathcal{H}(X),h)$

Agora que foram apresentadas as características básicas dos espaços métricos e de seus subconjuntos, pode-se definir o espaço métrico dos fractais. Esta definição é dividida em duas partes sendo a primeira a definição do espaço $\mathcal{H}$ e a segunda da métrica associada h.

Definição 2.9: "Seja (X,d) um espaço métrico completo. Então o espaço $\mathcal{H}(X)$ é o espaço dos subconjuntos compactos e não vazios do espaço X".

Desta forma, cada ponto $A \in \mathcal{H}(X)$ é um subconjunto compacto do espaço métrico (X,d), e, então, pode-se definir a distância entre um ponto $x \in X$ e um conjunto $A \in \mathcal{H}(X)$ como sendo:

$$d_1(x,A) = \min_{y \in A}\{d(x,y)\}. \tag{2.11}$$

Assim, a distância entre um ponto x e um conjunto compacto A é a distância entre o ponto x e o ponto $y \in A$ mais próximo dele. A Fig. 2.9 mostra dois exemplos do cálculo desta distância.

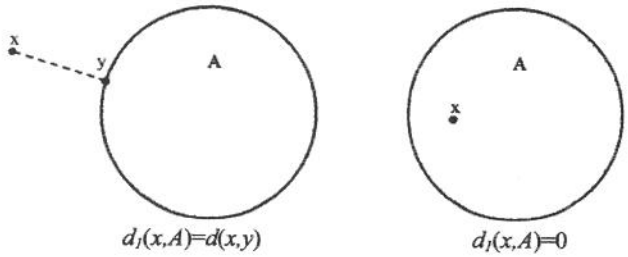
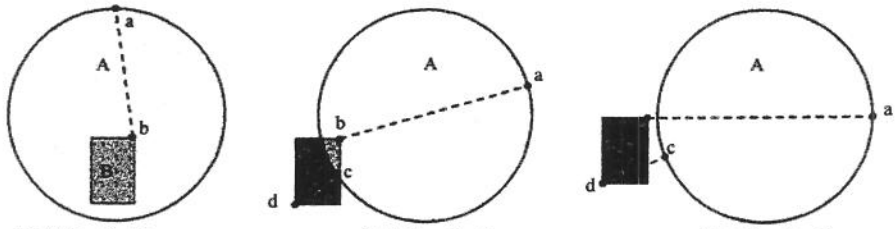


Fig. 2.9 – Exemplos da distância de um ponto a um conjunto

Note que como esta aplicação é definida por $d_1 : \mathbf{X}\times\mathcal{H}(X) \rightarrow \mathcal{R}$, não serve como métrica para nenhum dos espaços, mas pode ser usada como base para definir-se a aplicação d_2 abaixo:

$$d_2(A,B) = \max_{x\in A}\{d_1(x,B)\} = \max_{x\in A}\left\{ \min_{y\in B}\{d(x,y)\}\right\}. \tag{2.12}$$

Esta aplicação é um candidata a métrica pois $d_2 : \mathcal{H}(X)\times\mathcal{H}(X) \rightarrow \mathcal{R}$, no entanto, se tomarmos dois subconjuntos diferentes A e B, as distâncias $d_2(A,B)$ e $d_2(B,A)$ são diferentes como mostrado na Fig. 2.10 e, portanto, a distância d_2 não pode ser considerada uma métrica.



$d_2(A,B)=d(a,b)$
$d_2(B,A)=0$
$h(A,B)=d(a,b)$

$d_2(A,B)=d(a,b)$
$d_2(B,A)=d(d,c)$
$h(A,B)=d(a,b)$

$d_2(A,B)=d(a,b)$
$d_2(B,A)=d(d,c)$
$h(A,B)=d(a,b)$

Fig. 2.10 – Exemplos da distância entre conjuntos

Para contornar este problema define-se, então, a aplicação h: $\mathcal{H}(X)\times\mathcal{H}(X) \rightarrow \mathcal{R}$ tal que:

$$h = \max\{d_2(A,B), d_2(B,A)\} = \max\left\{ \max_{x\in A}\left\{ \min_{y\in B}\{d(x,y)\}\right\}, \max_{y\in B}\left\{ \min_{x\in A}\{d(x,y)\}\right\}\right\} \tag{2.13}$$

Pode-se provar [8] que a aplicação h é uma métrica, pois obedece às quatro propriedades requeridas para uma métrica. Esta aplicação h é chamada *métrica de Hausdorff* e é, em geral, a métrica utilizada no espaço dos fractais $\mathcal{H}(X)$. Note que a métrica de Hausdorff é dependente da métrica adotado no espaço (X,d).

O passo final para completar a definição do espaço dos fractais é dado pelo seguinte teorema [8] [10]:

Teorema 2.2: "Seja (X,d) um espaço métrico completo. Então $(\mathcal{H}(X),h)$ é um espaço métrico completo. Além disso, se $\{A_n \in \mathcal{H}(X)\}_{0<n<\infty}$ é uma seqüência de Cauchy, então:

$$\exists A = \lim_{n\rightarrow\infty} A_n \in \mathcal{H}(X) \tag{2.14}$$

que pode ser descrito como:

$$A = \{x \in X \mid \text{Há uma seqüência de cauchy } \{x_n \in A_n\} \text{ que converge para } x\}". \tag{2.15}$$

A seqüência de Cauchy para o espaço dos fractais é definida como [7] [8]:

Definição 2.10: "Uma seqüência $\{A_n : n=1, 2, ..., \infty\} \subset \mathcal{H}(X)$ é uma seqüência de Cauchy se, dado um $\varepsilon \in \mathcal{R} > 0$, existe um número inteiro positivo N tal que:

$$(A_n+\varepsilon \supset A_m \text{ e } A_m+\varepsilon \supset A_n) \Leftrightarrow h(A_m,A_m) \leq \varepsilon, \quad \forall m,n>N \tag{2.16}$$

onde: $A+\varepsilon = \{y \in X \mid d(x,y) \leq \varepsilon \text{ para algum } x \in A\}$".

Definido, assim, o espaço onde os fractais existem, pode-se então aprofundar-se no processo de geração dos fractais. Esse processo desenvolvido originalmente por Mandelbrot [13] envolve a utilização de transformações definidas no espaço métrico. Assim, a próxima seção apresentará em resumo das características deste tipo de transformações.

2.8 - Transformações em espaços métricos

A geometria fractal estuda subconjuntos "complicados" de espaços geometricamente simples como $\mathcal{R}$ ou $\mathcal{R}^2$. A geometria fractal determinística enfoca os subconjuntos do espaço que são gerados por transformações simples do espaço sobre si mesmo, ou que possuam propriedades invariantes sob estas transformações. Estas transformações simples são, em geral, completamente especificadas por um pequeno número de parâmetros. Como exemplo pode-se citar as transformações afins que, em $\mathcal{R}^2$, são completamente especificadas por uma matriz 2x2 e um vetor com 2 posições totalizando 6 parâmetros.

Primeiramente, uma transformação é uma aplicação de um espaço métrico X em si mesmo é definida matematicamente por [8] [9]:

Definição 2.11: "Seja (X,d) um espaço métrico. Uma transformação em X é uma função $f: X \rightarrow X$ que associa exatamente um ponto $f(x) \in X$ a cada ponto $x \in X$".

Além disso, se f estabelece uma relação um-a-um entre $f(x)$ e x de forma que $f(x)=f(y) \Leftrightarrow x=y$, então é possível definir uma função $f^{-1} : X \rightarrow X$ tal que $f(f^{-1}(x))=x$. Neste caso a função f é chamada inversível e a função f^{-1} é a *função inversa de* f.

Como estas funções foram definidas do espaço métrico sobre si mesmo, então pode-se definir as aplicações sucessivas (iterações) da função f como sendo:

Definição 2.12: "Seja $f: X \rightarrow X$ uma transformação em um espaço métrico. As *iterações diretas* de f são as transformações $f^{\circ n} : X \rightarrow X$ definido por:

$$f^{\circ 0}(x)=x, f^{\circ 1}(x)=f(x), f^{\circ 2}(x)=f(f(x)),..., f^{\circ n}(x)= f(f^{\circ (n-1)}(x)) \text{ para } n=0, 1, 2, ... \tag{2.17}$$

Se f for inversível, então as *iterações reversas* de f são as transformações $f^{o(-m)}$: $X \rightarrow X$:

$$f^{o(-1)}(x)=f^{-1}(x), f^{o(-2)}(x)=(f^2(x))^{-1},..., f^{o(-m)}(x)=(f^m(x))^{-1} \text{ para } m=1, 2, 3, \ldots . \quad (2.18)$$

Dois exemplos de transformações no espaço $\mathfrak{R}$ são as transformações polinomiais e as transformações fracionais lineares ou transformações de Möbius definidas a seguir:

Definição 2.13: "Uma Transformação Polinomial de Grau N f: $\mathfrak{R} \rightarrow \mathfrak{R}$ é expressa na forma:

$$f(x)= a_0 + a_1.x + a_2.x + \ldots + a_N.x \quad (2.19)$$

onde os coeficientes a_i, $i=0, 1, \ldots, N$, são números reais com $a_N \neq 0$ e N é um inteiro não-negativo".

Definição 2.14: "Uma Transformação Fracional Linear (ou Transformação de Möbius) f: $\mathfrak{R} \rightarrow \mathfrak{R}$ é expressa na forma:

$$f(x)=\frac{a \cdot x+b}{c \cdot x+d}, \quad a,b,c,d \in \mathfrak{R}, ad \neq bc". \quad (2.20)$$

Dentro das transformações polinomiais existe uma família de transformações que são chamadas *transformações afins*. As transformações afins são transformações polinomiais de grau 1, ou seja, na forma: $f(x)=a.x+b$, que são muito importantes para os esquemas de compressão fractal de imagens. No próximo item serão apresentados detalhes sobre esta família de transformações no espaço métrico $(\mathfrak{R}^2, d_{L2})$.

Transformações Afins no Plano Euclidiano

Como uma extensão do caso unidimensional, pode-se definir matematicamente uma transformação afim bidimensional por:

Definição 2.15: "Uma transformação afim ω: $\mathfrak{R}^2 \rightarrow \mathfrak{R}^2$ é uma transformação do tipo:

$$\omega(x_1,x_2)=(ax_1+bx_2+c, dx_1+ex_2+f), \text{ onde } a, b, c, d, e, f \in \mathfrak{R} \quad (2.21)$$

e pode ser expressa na forma matricial:

$$\omega(x)=\omega\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}=\begin{pmatrix} a & b \\ d & e \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}+\begin{pmatrix} c \\ f \end{pmatrix}=\mathbf{A}x+\mathbf{t}". \quad (2.22)$$

Note que a matriz A sempre pode ser representada na forma polar:

$$\begin{pmatrix} a & b \\ d & e \end{pmatrix}=\begin{pmatrix} r_1 \cdot sen\theta_1 & -r_2 \cdot sen\theta_2 \\ r_1 \cdot cos\theta_1 & r_2 \cdot cos\theta_2 \end{pmatrix} \quad (2.23)$$

onde os pares (r_1, θ_1) e $(r_2, \theta_2+\pi)$ são as representações polares dos pares (a,d) e (b,e) respectivamente. Assim, a transformação linear em $\mathfrak{R}^2$:

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \mathbf{A} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad (2.24)$$

mapeia qualquer polígono com um vértice na origem para outro polígono com um vértice na origem como nos exemplos mostrados na Fig. 2.11. Note que o polígono pode ser invertido lateralmente como na Fig. 2.11b (note o desenho no centro do triângulo).

Assim, uma transformação afim genérica $\omega(x)=Ax+t$ em $\mathfrak{R}^2$ consiste em uma transformação linear A que deforma o espaço com relação à origem (e sem alterar a origem do espaço) seguida de uma translação ou deslocamento especificado pelo vetor $t=(e,f)$ como mostrado na Fig.2.12.

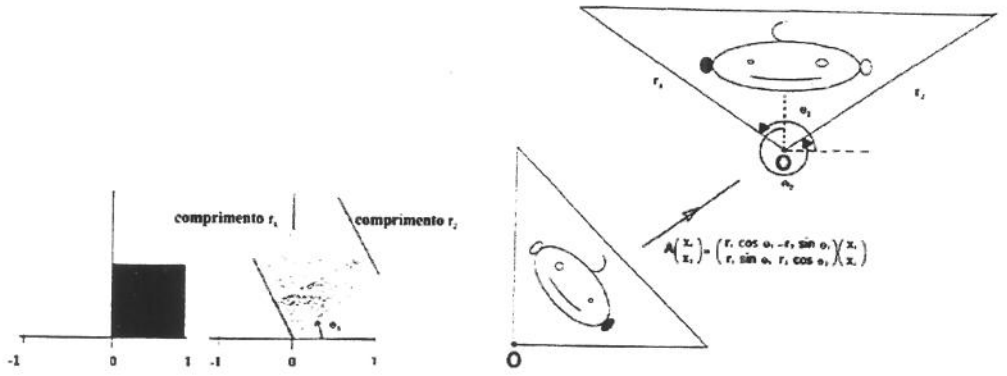


Fig. 2.11 – Exemplos de transformações $x \rightarrow A.x$: a) Sem inversão lateral; b) Com inversão lateral .

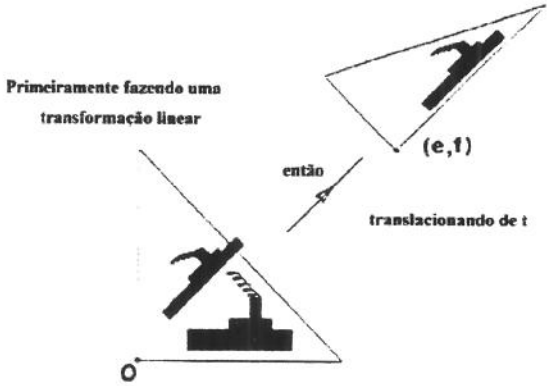


Fig. 2.12 – Transformação Afim $x \rightarrow A.x + t$.

Quando uma transformação afim tiver $|r_1|=|r_2|=r$ e $\theta_1=\theta_2=\theta$, esta transformação é chamada de *similitude* e então, θ é chamado ângulo de rotação e r é o fator de escala ou escalonamento. Uma característica nas similitudes é que elas preservam os ângulos do objeto transformado.

2.9 - Teorema do Mapeamento Contrativo

Antes de enunciar o Teorema do Mapeamento Contrativo é necessário definir o conceito de contratividade [10]:

Definição 2.18: "Uma transformação $f: X \to X$ em um espaço métrico (X,d) é chamada *contrativa* ou *mapeamento contrativo* se há uma constante real $0 \leq s < 1$ tal que:

$$d(f(x),f(y)) \leq s.d(x,y), \ \forall x,y \in X, \tag{2.25}$$

onde o número s é chamado fator de contratividade de f".

Teorema 2.3: "Seja a transformação $f: X \to X$ um mapeamento contrativo em um espaço métrico completo (X,d), então há um ponto $x_f \in X$, chamado ponto fixo de f, tal que, para qualquer ponto $x \in X$, a seqüência $\{f^{on}(x) \mid n=0,1,2,...\}$ converge para o ponto x_f. Ou seja:

$$\lim_{n\to\infty} f^{on}(x) = x_f, \quad \forall x \in X \text{"}. \tag{2.26}$$

Corolário – Teorema de Colagem: "Seja a transformação $f: X \to X$ um mapeamento contrativo em um espaço métrico completo (X,d) com um ponto fixo x_f, então:

$$d(x, x_f) \leq \frac{1}{1-s} \cdot d(x, f(x)), \ \forall x \in X \text{"}. \tag{2.27}$$

Como este teorema e seu corolário são de importância fundamental para o processo de geração de fractais e, por conseguinte, para os esquemas de compressão de imagens utilizando fractais, suas provas são apresentadas a seguir.

Prova do Teorema 2.3: "Seja $x \in X$ e um número real $s \in [0,1)$ o fator de contratividade do mapeamento contrativo f. Para $\forall\ m > n \geq 0$:

$$d(f^{on}(x), f^{om}(x)) \leq s \cdot d(f^{o(n-1)}(x), f^{o(m-1)}(x)) \leq s^2 \cdot d(f^{o(n-2)}(x), f^{o(m-2)}(x)) \leq s^n \cdot d(x, f^{o(m-n)}(x))$$

$$d(f^{on}(x), f^{om}(x)) \leq s^n \cdot [d(x, f(x)) + d(f(x), f^{o2}(x)) + ... + d(f^{o(m-n-1)}(x), f^{o(m-n)}(x))]$$

$$d(f^{on}(x), f^{om}(x)) \leq s^n \cdot \sum_{k=1}^{m-n} d(f^{o(k-1)}(x), f^{ok}(x))$$

$$d(f^{on}(x), f^{om}(x)) \leq s^n \cdot \sum_{k=1}^{m-n} s^{k-1} \cdot d(x, f(x))$$

$$d(f^{on}(x), f^{om}(x)) \leq \frac{s^n}{1-s} \cdot d(x, f(x)) \tag{2.28}$$

Portanto a seqüência $\{f^{on}(x) \mid n=0,1,2,....\}$ é uma seqüência de Cauchy. Como o espaço X é um espaço completo, há um ponto tal que:

$$\lim_{n\to\infty} f^{on}(x) = x_f \Rightarrow f(x_f) = f(\lim_{n\to\infty} f^{on}(x_f)) = \lim_{n\to\infty} f^{o(n+1)}(x_f) = x_f \tag{2.29}$$

o que garante a existência do ponto fixo. Para provar a unicidade deste ponto fixo, suponha que haja dois pontos fixos x_f e y_f. Portanto $x_f = f(x_f)$ e $y_f = f(y_f)$ e então:

$$d(x_f, y_f) \le d(f(x_f), f(y_f)) \le s \cdot d(x_f, y_f),$$

$$(1-s)\, d(x_f, y_f) \le 0. \tag{2.30}$$

Como $0 \le s < 1$, a condição acima só é obedecida se $d(x_f, y_f) = 0$ e, portanto, $x_f = y_f$ provando assim a unicidade do ponto fixo. A prova do Teorema de Colagem é feita tomando $n=0$ e $m \to \infty$ na equação (2.27).

Mapeamentos Contrativos no Espaço dos Fractais

Até agora foram definidos mapeamentos contrativos em espaços métricos genéricos e só foram apresentados exemplos de mapeamentos para espaços geometricamente simples. No entanto o espaço métrico $(\mathcal{H}(\mathbf{X}), h)$ não é geometricamente simples e a sua métrica é complexa, o que faz com que a definição dos mapeamentos contrativos e a análise da convergência de seqüências sejam consideravelmente mais complexas.

Para contornar este problema em [7] apresenta-se o seguinte Lema que analisa a relação entre um mapeamento contrativo $\omega : X \to X$ e o seu equivalente no espaço $(\mathcal{H}(X), h)$.

Lema 2.1: "Seja $\omega : X \to X$ um mapeamento contrativo no espaço métrico (X,d) com fator de contratividade s. Então o mapeamento $\omega : \mathcal{H}(X) \to \mathcal{H}(X)$ definido por:

$$\omega(B) = \{\omega(x) \mid x \in B\}, \quad \forall B \in \mathcal{H}(X) \tag{2.31}$$

é um mapeamento contrativo em $(\mathcal{H}(X), h)$ com fator de contratividade s ".

Prova do Lema 2.1: "Como ω é uma mapeamento contrativo, então ele é contínuo [12]. Assim, tomando-se um subconjunto $B \subset X$ não-vazio, $\omega(B) = \{\omega(x) \mid x \in B\}$ é não vazio. Se B for compacto, então pode-se provar [12] que $\omega(B)$ também é compacto (pois a Compacidade é um invariante topológico), ou seja, a função ω definida desta forma mapeia o espaço $\mathcal{H}(X)$ para si mesmo.

Desta forma, sejam $B, C \in \mathcal{H}(X)$, então:

$$h(\omega(B), \omega(C)) = \max\left\{\max_{x\in B}\left\{\min_{y\in C}\{d(\omega(x), \omega(y))\}\right\}, \max_{y\in C}\left\{\min_{x\in B}\{d(\omega(x), \omega(y))\}\right\}\right\}$$

$$h(\omega(B), \omega(C)) \le \max\left\{\max_{x\in B}\left\{\min_{y\in C}\{s \cdot d(x,y)\}\right\}, \max_{y\in C}\left\{\min_{x\in B}\{s \cdot d(x,y)\}\right\}\right\} = s \cdot h(B,C) \tag{2.32}$$

portanto, $\omega : \mathcal{H}(X) \rightarrow \mathcal{H}(X)$ é um mapeamento contrativos com fator de contratividade s ".

Assim, para determinar se um mapa é contrativo no espaço de fractais $\mathcal{H}(X)$, basta determinar a sua contratividade no espaço X, o que é bem mais fácil devido ao fato deste espaço ser, em geral, geometricamente simples.

2.10 - Conclusões

Neste capítulo foi apresentada a introdução teórica dos princípios matemáticos necessários à compreensão da Teoria de Fractais. Foram abordados os conceitos de espaços métricos, propriedades de subconjuntos de espaços, e transformações contrativas. Baseando-se nesses princípios, no próximo capítulo serão introduzidos o Sistema de Funções Iterativas, que constitui a principal ferramenta para a construção de fractais, e a sua variante Sistema Particionado de Funções Iterativas que é a base dos codificadores fractais de imagem existentes atualmente.

Capítulo 3

Construção de Fractais: Sistema de Funções Iterativas – IFS

3.1 - Sistemas de Funções Iterativas (IFS – Iterated Function Systems)

No capítulo anterior, apresentou-se o conceito de contratividade para um mapeamento simples no espaço de fractais $\mathcal{H}(X)$. No entanto esta definição pode ser entendida para conjuntos de mapeamentos contrativos de forma a possibilitar outras maneiras de geração de curvas fractais. O Lema a seguir, cuja prova é apresentada em [8], define a contratividade de conjuntos de mapeamentos contrativos.

Lema 3.1: "Seja (X, d) um espaço métrico completo e $\{\omega_n : X \to X, n=1, 2, ..., N\}$ um conjunto de mapeamentos contrativos neste espaço, e portanto, em $(\mathcal{H}(X), h)$. Seja o fator de contratividade de ω_n dado por s_n para cada n. Definindo $W : \mathcal{H}(X) \to \mathcal{H}(X)$ por:

$$W(B) = \omega_1(B) \cup \omega_2(B) \cup ... \cup \omega_N(B) \tag{3.1}$$

$$W(B) = \bigcup_{n=1}^{N} \omega_n(B), \text{ para cada } B \in \mathcal{H}(X). \tag{3.2}$$

então W é um mapeamento contrativo com fator de contratividade $s = max \{s_n, n=1, 2, ..., N\}$".

Com este Lema define-se os Sistemas de Funções Iterativas (IFS – *Iterated Function Systems*) :

Definição 3.1: "Um Sistema de Funções Iterativas (hiperbólicas) – IFS – consiste em um espaço métrico completo (X,d) e um conjunto finito de mapeamentos contrativos $\omega_n : X \to X$, com os respectivos fatores de contratividade s_n, para $n=1, 2, ..., N$. A notação para a IFS é $\{X; \omega_n, n=1, 2, ..., N\}$ e seu fator de contratividade é $s = max \{s_n, n=1, 2, ..., N\}$".

A palavra "hiperbólicas" foi colocada entre parênteses porque, na prática, ela é quase sempre omitida e o sistema é referenciado pela sigla IFS. O teorema a seguir resume as principais propriedades dos sistemas de funções iterativas e define o conceito de *atrator* de um IFS.

Teorema 3.1: "Seja $\{X;\ \omega_n,\ n=1,\ 2,\ ...,\ N\}$ um sistema de funções iterativas com fator de contratividade s. Então a transformação $W : \mathcal{H}(X) \rightarrow \mathcal{H}(X)$ é definida por:

$$W(B) = \bigcup_{n=1}^{N} \omega_n(B) \tag{3.3}$$

para todo $B \in \mathcal{H}(X)$, é um mapeamento contrativo no espaço métrico completo $(\mathcal{H}(X), h)$ com fator de contratividade s, ou seja:

$$h(W(B), W(C)) \leq s \cdot h(B, C), \tag{3.4}$$

para todo $B, C \in \mathcal{H}(X)$. Seu único ponto fixo, $A \in \mathcal{H}(X)$ e chamado de *atrator* da IFS, é dado por (Teorema do Ponto Fixo de Mapas Contrativos):

$$A = \lim_{n \to \infty} W^{on}(B), \quad \forall B \in \mathcal{H}(X), \tag{3.5}$$

de forma que $A = W(A) = \bigcup_{n=1}^{N} \omega_n(A)$ e a distância entre um dado $B \in \mathcal{H}(X)$ e o atrator A obedece à inequação (Teorema da Colagem):

$$h(B, A) \leq \frac{1}{1-s} h(B, W(B))". \tag{3.6}$$

Este teorema resume as características que possibilitam o modelamento de fractais como sendo subconjuntos compactos de um espaço métrico completo que pode ser completamente especificado através de um sistema de equações (mapas contrativos). Neste ponto, pode-se introduzir, então, um conceito um pouco mais refinado de fractal.

Fractais: características e definições

Como foi dito anteriormente, os fractais não possuem uma definição formal aceita universalmente. No entanto, se um subconjunto F é um fractal, então ele deve apresentar as (ou pelo menos algumas das) características gerais a seguir [7]:

1. Há uma descrição algorítmica simples de F;
2. A *"Dimensão Fractal"* de F é maior do que sua dimensão topológica (definidas em [7][14]);
3. F tem detalhes em todas as escalas;
4. F é (exatamente ou aproximadamente) auto-similar.

A descrição algorítmica referenciada na característica 1 pode ser vista, basicamente, como os sistemas de funções iterativas apresentados anteriormente. Os IFS são baseados em mapas

contrativos que são, em geral, relativamente simples. Um método mais geral (e mais rápido) de construção de fractais é o método desenvolvido por Barnsley, Elton e Hardin [2] chamado de Sistema Recorrente de Funções Iterativas (RIFS – *Recurrent Iterated Function System*).

Como as características de Dimensão Fractal e dimensão topológica envolvem muitos outros conceitos de Topologia, sua apresentação aqui foge ao escopo da tese e, portanto, uma descrição mais detalhada será apenas referenciada [7][14].

As outras duas características, em especial o conceito de auto-similaridade, serão comentadas no próximo item.

Auto-similaridade em fractais

Considere um fractal gerado pelo processo aplicações recursivas de um conjunto de transformações afins contrativas no espaço $\mathcal{H}(X)$. A característica comum a todos os atratores frutos desse tipo de transformação é que eles são gerados por infinitas reduções, rotações e translações de uma imagem qualquer. Isto implica que ao ampliarmos uma das regiões de um atrator, ver-se-á cópias dele, ou seja, cada atrator é formado por cópias reduzidas e transformadas de si mesmo. Assim, em todas as escalas, o nível de detalhamento é rigorosamente igual ao nível de detalhamento da imagem original fazendo, assim, com que um fractal possua detalhes em todas as escalas.

Essa característica dos fractais é chamada *auto-similaridade* que, juntamente com o Teorema de Colagem, formam a base dos esquemas de compressão fractal de imagens. Esse método de gerar fractais foi criado por John Hutchinson [1].

No entanto, ao tratarmos com imagens naturais, pode-se facilmente notar que estas, no caso geral, não são exatamente auto-similares, ou seja, não são formadas por cópias reduzidas e transformadas de si mesmas. Ainda assim, as imagens naturais podem conter auto-similaridade "por partes", ou seja, uma parte da imagem pode se assemelhar a outra parte da mesma imagem, como é mostrado na seção a seguir.

3.2 - Auto-similaridade em imagens

Para exemplificar o tipo de auto-similaridade existente nas imagens naturais, considere a imagem *Lena* (Fig. 3.1a). Pode-se observar que ela não contém o tipo de auto-similaridade normalmente encontrada nos fractais, mas na verdade esta imagem contém um tipo diferente de auto-similaridade. A Fig. 3.1b destaca algumas regiões desta imagem que são similares em diferentes escalas.

a) b)

Fig. 3.1 – Imagem Lena: a) Original; b) Exemplos de similaridades dessa imagem.

A diferença entre um fractal formado por um IFS e esta imagem é que esta é formada por cópias transformadas de partes de si mesma, e não da imagem inteira. Estas partes transformadas, em geral, não se encaixam perfeitamente na composição da imagem final exata de forma que é necessário que se permita algum erro na representação de uma imagem por um conjunto de transformações afins. Isto significa que a imagem codificada será apenas uma aproximação da imagem original.

Assim, para modelar uma imagem natural por um sistema de funções iterativas, é necessário particionar a imagem de forma que cada parte da imagem original (chamado bloco-imagem ou *range block*) seja similar com outra parte da imagem (chamado bloco-domínio ou *domain block*) e possa, assim, ser gerada através de aplicações sucessivas de uma IFS. A partir desta idéia Jacquin [3] desenvolveu o Sistema Particionado de Funções Iterativas (PIFS – *Partitioned Iterated Functions Systems*).

3.3 - Sistema Particionado de Funções Iterativas (PIFS)

A grande diferença entre o IFS e o PIFS é que no primeiro o mapa contrativo é aplicado obrigatoriamente sobre a imagem toda (o domínio da aplicação é o espaço inteiro). Já no caso do PIFS o domínio da aplicação pode ser qualquer subconjunto compacto do espaço. Ou seja, o PIFS pode ser visto como uma generalização do IFS.

Esta generalização simplifica (e torna possível na maioria dos casos) a codificação de conjuntos não auto-similares, como as imagens naturais. Uma definição mais formal do PIFS é [7]:

Definição 3.2: "Seja (X,d) um espaço métrico completo, e D_i, $R_i \subset X$, para $i = 1, 2, ..., n$ tal que:

$$\bigcup_{i=1}^{n} R_i = I^2 . \tag{3.7}$$

onde I^2 representa o quadrado unitário. Um sistema particionado de funções iterativas é uma coleção de mapas contrativos, $W=\{\omega_i : D_i \rightarrow R_i \mid i=1, 2, ..., n\}$".

Seria interessante neste ponto utilizar o Teorema do Ponto Fixo de Mapeamento Contrativo para provar a existência e a unicidade do ponto fixo para um PIFS. No entanto, segundo Fisher [10], esta prova é impossível no sentido completamente geral.

Um dos problemas é que, sendo os domínios restritos, no caso geral, a escolha do ponto inicial se torna relevante e, se não for escolhido apropriadamente, as aplicações recursivas podem levar a um conjunto vazio (que não pertence ao espaço de fractais). No entanto, no caso específico da representação de imagens naturais, este problema não ocorre. Apesar da teoria completa sobre o PIFS ainda estar por ser desenvolvida, muitos casos especiais já foram estudados, como o caso da compressão de imagens, o qual será apresentado na próxima seção.

Uma codificação simples de imagens com PIFS

Apesar de ser um pouco restritiva no sentido geral, neste item será utilizada a mesma notação já utilizada para descrever o IFS, com a finalidade de enfatizar a aplicação do PIFS em uma imagem natural e não o princípio matemático por trás do sistema.

O primeiro passo é a definição do modelo matemático que será usado para a imagem natural:

Definição 3.3: "Seja $I = [0;1] \subset \mathfrak{R}$ o intervalo unitário na reta real e, conseqüentemente, I^2 o quadrado unitário no plano euclidiano. Uma imagem natural pode ser vista como o gráfico de uma função $f \in \boldsymbol{F}$ tal que $\boldsymbol{F} = \{f \mid f : I^2 \rightarrow \mathfrak{R}\}$".

Deve-se notar que a função f mapeia o quadrado unitário para o intervalo unitário, mas considera-se que a imagem de f seja $\mathfrak{R}$ para que as somas e subtrações de imagens fiquem bem definidas. Adotando a métrica do supremo d_{sup} para este espaço, Rudin [15] provou que o espaço métrico $(\boldsymbol{F}, d_{sup})$ é um espaço métrico completo. Embora a imagem e o domínio da função sejam, respectivamente, R_i x I e D_i x I, é usual referenciá-los apenas como bloco-imagem R_i e bloco-domínio D_i.

Definido o modelo de imagem, suponha que se deseja codificar por fractal (PIFS) uma imagem f. Ou seja, deseja-se encontrar a coleção de mapas $\omega_1, \omega_2, ..., \omega_n$ com:

$$\omega_i\begin{bmatrix}x\\y\\z\end{bmatrix}=\begin{bmatrix}a_i & b_i & 0\\c_i & d_i & 0\\0 & 0 & s_i\end{bmatrix}\cdot\begin{bmatrix}x\\y\\z\end{bmatrix}+\begin{bmatrix}e_i\\f_i\\o_i\end{bmatrix},\ i=1,\ 2,\ ...,\ n \tag{3.8}$$

e com $W(f)$ definido por:

$$W(f)=\bigcup_{i=1}^{N}\omega_i(f\cap(D_i\times I)) \tag{3.9}$$

de modo que $f=f_W$ seja o atrator de W. Usando a notação $\omega_i(f)$ para indicar $\omega_i(f\cap(D_i\times I))$, a equação do ponto fixo pode ser expressa como:

$$f_w=W(f_w)=\omega_1(f_w)\cup\omega_2(f_w)\cup...\,\omega_n(f_w). \tag{3.10}$$

Esta equação sugere como pode ser realizada a codificação: procura-se uma partição de f que, submetida à transformação W resulte novamente em f. Em geral, esta condição não pode ser alcançada completamente, visto que as imagens naturais não são formadas por pedaços que podem ser transformados para encaixar em outro lugar da imagem. Assim, o objetivo, na prática, é encontrar $f'=f_W$ tal que o erro $d_{sup}(f,f')$ seja pequeno o suficiente. Neste caso:

$$f\approx f'=W(f')\approx W(f)=\omega_1(f)\cup\omega_2(f)\cup...\,\omega_n(f) \tag{3.11}$$

Então, é suficiente aproximar as partes da imagem das partes transformadas minimizando a expressão:

$$\min_{\omega_i}\{d_{sup}(f\cap(R_i\times I),\omega_i(f))\},\quad i=1,\ ...,\ n \tag{3.12}$$

Assim, dada uma imagem particionada em regiões R_i, devem-se achar as partes D_i e os mapas ω_i tais que, ao aplicar-se ω_i à parte da imagem sobre D_i, obtém-se um resultado muito próximo à parte da imagem sobre R_i.

Para facilitar a codificação, os mapas ω_i são, em geral, selecionados a partir de um conjunto limitado de possíveis mapas. Analisando a forma geral dos mapas ω_i (equação (3.7)) pode-se dividir cada mapa em duas partes: A) Parte geométrica composta pelos coeficientes: a_i, b_i, c_i, d_i, e_i, f_i que escalonam, rotacionam e transladam o bloco-domínio sobre o bloco-imagem; e B) Ajuste de brilho/contraste: formado pelos coeficientes s_i e o_i.

Quando se utiliza partição *quadtree* ou *H-V*, o formato dos blocos-imagem e dos blocos-domínio é sempre retangular (como será mostrado no próximo item) e o tamanho do bloco-domínio é escolhido, em geral, o dobro do tamanho do bloco-imagem. Assim, a escolha da parte geométrica do mapa é limitada a apenas uma redução de tamanho (subamostragem) por um fator de 2 seguida de uma dentre as 8 opções: 4 rotações (0°, 90°, 180° e 270°) e 4 inversões (vertical, horizontal, diagonal principal, diagonal secundária). Assim, a minimização da equação (3.11) fica bastante

simplificada, bastando encontrar os valores s_i e o_i que minimizam a distância em cada uma das 8 transformações geométricas possíveis.

Para concluir a descrição do processo de codificação ainda é necessário definir como é feita a partição da imagem em blocos-imagem R_i e como se encontram os blocos-domínio correspondentes.

Particionamento da imagem: blocos-imagem e blocos domínio

O modo mais simples de se particionar uma imagem é dividi-la em blocos de mesmo tamanho, como é feito nos esquemas tradicionais de compressão de imagem. No entanto, esta forma de particionar não é boa para a codificação fractal, pois há regiões que possuem um grande número de detalhes, o que torna difícil a obtenção de um mapa ω_i (e de uma parte associada D_i) que resultem em um erro pequeno o suficiente. Similarmente, há regiões grandes e com poucos detalhes que possibilitam a utilização de um mesmo mapa, de forma que a codificação final fique mais eficiente.

Assim, uma generalização da partição de tamanho fixo é a *Partição Quadtree*. Nesse esquema de particionamento, se em uma dada região R_i não for possível encontrar um D_i tal que o erro (3.2) não for menor que um erro máximo estipulado, então se subdivide essa região em quatro partes iguais processando cada uma em separado. Esse algoritmo é recursivo, ou seja, se qualquer uma dessas subdivisões também não alcançar o erro desejado, então ela é subdividida em 4 partes e recomeça-se o processo. O processo de particionamento básico inicia com a imagem inteira e segue recursivamente até que todas as partições atinjam o erro desejado, mas é comum a especificação de um particionamento inicial mínimo de forma a limitar o máximo tamanho dos blocos-imagem.

O Conjunto Domínio $\boldsymbol{D}$ (conjunto de todos os possíveis blocos-domínio) pode ser escolhido como sendo todos os possíveis quadrados dentro da imagem, no entanto, é comum [7] escolher-se $\boldsymbol{D}$ como um subconjunto deste espaço. Um exemplo de conjunto domínio [16] é o conjunto de quadrados de tamanho fixo centrados numa grade com espaçamento de metade do tamanho de cada D_i. É conveniente [10] e muito usual a escolha do tamanho do bloco como sendo o dobro do tamanho do cada bloco-imagem R_i.

A desvantagem da utilização da partição *quadtree* é que na determinação do conjunto domínio $\boldsymbol{D}$ não se leva em conta o contexto da imagem fazendo com que a coleção de possíveis blocos-domínio deva ser bastante grande para que se encontre uma boa aproximação para um dado bloco-imagem R_i.

Uma forma de contornar esse problema é aumentar a flexibilidade da partição da região

usando a partição *H-V*. Nesse esquema de partição, a imagem é particionada horizontal ou verticalmente para formar dois novos retângulos. Como a posição da partição é flexível, pode-se fazer as partições de forma a tentar explorar a autossimilaridade das imagens como, por exemplo, escolher as partições tal que as bordas das imagens tendam a ficar diagonalmente dentro de um dos retângulos resultantes (Fig. 3.2).

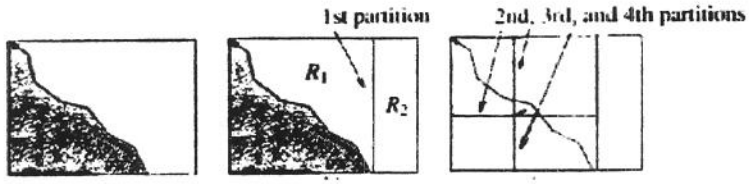
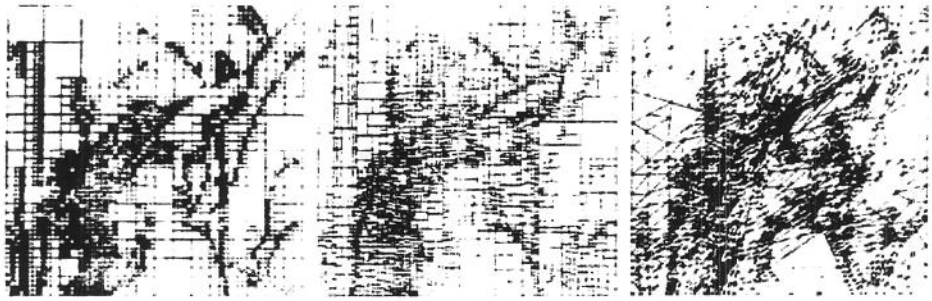


Fig. 3.2 – Esquema de partição *H-V* para explorar auto-similaridade.

O particionamento é recursivo, como no esquema *quadtree*, até que o erro mínimo seja atingido em cada região. Como o particionamento é adaptativo em cada bloco neste esquema, além do aumento na complexidade computacional do codificador (e do decodificador), também torna-se necessário o envio da posição de quebra, aumentando a quantidade de dados a ser codificada, o que prejudica um pouco o desempenho geral do esquema.

Há ainda muitos outros tipos de particionamento que podem explorar diferentes características e formatos, como a partição triangular mostrada na Fig. 3.3, apresentando resultados melhores em termos de qualidade e, em geral, um grande aumento da complexidade computacional do codificador e decodificador.



(a) (b) (c)

Fig. 3.3 – Exemplos de partição de imagens. (a) *Quadtree*; (b) *H-V*; (c) Triangular.

No entanto, a determinação do tipo de partição a ser utilizada em um esquema de codificação fractal não deve levar em conta somente o critério da qualidade, e sim o critério de taxa-distorção, pois quanto mais complexa a partição, mais informação deve ser transmitida entre o codificador e o decodificador, reduzindo, assim, a taxa de compressão final.

Nos artigos publicados recentemente nas revistas científicas, praticamente a totalidade utiliza a partição *quadtree* pela sua simplicidade e boas curvas de taxa-distorção quando usadas imagens naturais.

Exemplo de codificadores PIFS

Como os esquemas de partição são em geral adaptativos, como o *quadtree* e o *H-V*, o número de blocos-imagem não é fixo e, quanto maior o número de blocos-imagem, maior a fidelidade, mais transformações e, conseqüentemente, menor a compressão. Este balanceamento entre fidelidade e taxa de compressão leva a duas possíveis aproximações de codificação: visando uma determinada compressão ou uma determinada fidelidade. O fluxograma de um esquema básico buscando uma fidelidade mínima e_c é mostrador na Fig. 3.4 [7]. Sendo que $tam(R_i)$ se refere ao tamanho do bloco-imagem R_i e r_{min} é o parâmetro que designa o tamanho mínimo permitido ao bloco-imagem R_i.

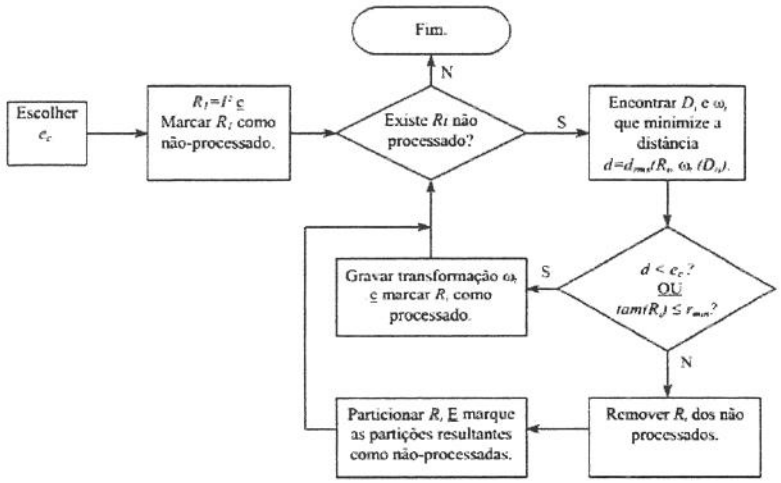


Fig. 3.4 – Fluxograma do Codificador Fractal de Imagens com Fidelidade Almejada.

Pode-se notar que apesar de o parâmetro e_c ser a fidelidade mínima para cada bloco-imagem, este valor não garante que a imagem codificada tenha esta fidelidade, pois a fidelidade depende também do valor de r_{min} e do conjunto domínio. No entanto, quanto menor o e_c, melhor a fidelidade da imagem codificada.

Na Fig. 3.5 é mostrado o codificador fractal buscando uma determinada taxa de compressão (através da determinação do número máximo de transformações). Novamente, a taxa de compressão almejada não é exatamente alcançada, mas as variações tendem a ser pequenas o suficiente para que se tenha um bom grau de precisão na especificação da taxa de compressão.

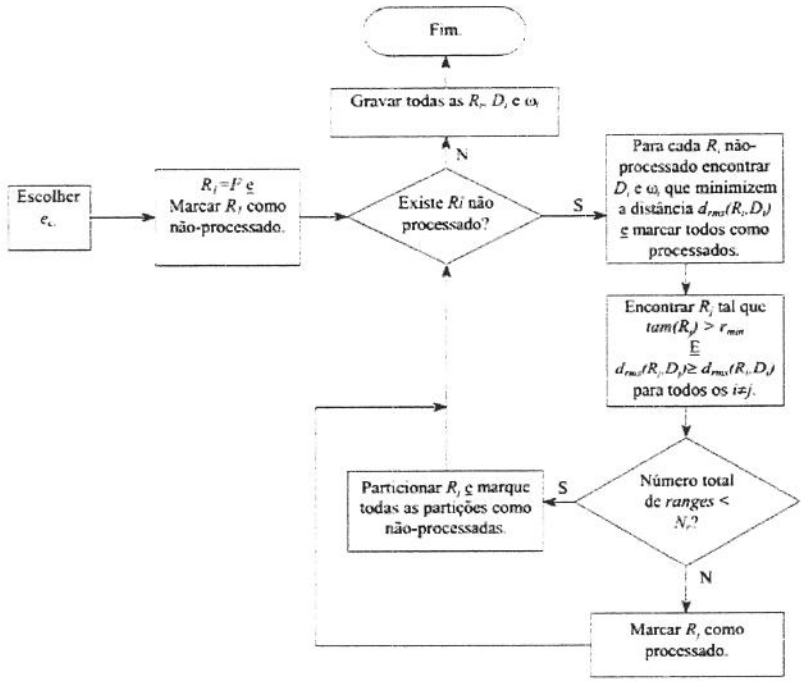


Fig. 3.5 – Fluxograma do Codificador Fractal de Imagens com Taxa de Compressão Almejada.

Para ambos os esquemas, o processo de decodificação é igual e relativamente simples: basta pegar-se uma imagem inicial qualquer e, para cada D_i designado no código, aplica-se a transformação ω_i e coloca-se o resultado na posição R_i. Aplica-se recursivamente estas transformações até atingir o ponto fixo. Tipicamente são necessárias em torno de 5 iterações para decodificar uma imagem e, em alguns casos em particular, é possível decodificar-se a imagem usando um número exato de iterações. Utilizando-se um máximo de 10 iterações a convergência é garantida praticamente na totalidade dos casos.

3.4 - Conclusões

Nesse capítulo foram apresentados o método de geração de fractais chamado Sistema de Funções Iterativas (IFS) e a sua adaptação à codificação de imagens digitais: Sistema Particionado de Funções Iterativas (PIFS), incluindo as descrições dos sistemas de partição *quadtree* e *H-V* que formam a base de praticamente a totalidade dos codificadores fractais de imagens encontrados na literatura recente. Nos próximos capítulos serão apresentados os detalhes de alguns codificadores propostos e que serviram de base de comparação para o algoritmo proposto nessa tese.

Capítulo 4

Codificadores Fractais de Imagem

4.1 - Introdução

Neste capítulo serão apresentados dois esquemas de codificação fractal, o primeiro proposto por Fisher em [7] que implementa um esquema de classificação de blocos para acelerar a procura pelo casamento entre blocos de um sistema PIFS utilizando partição *quadtree*. Apesar de os resultados obtidos não serem otimizados, este esquema serve como ponto de referência para muitos outros esquemas fractais por apresentar um bom compromisso entre complexidade, qualidade e tempo de codificação.

O segundo esquema foi proposto por Cardinal [17] e se trata de uma modificação da classificação por vetor de características originalmente proposta por Saupe. Este codificador foi escolhido por ter sido recomendado como base de comparação pelos revisores da *IEEE Transactions on Image Processing* durante o processo de revisão do artigo que publicamos em 2005.

Em ambos os sistemas, optou-se por trabalhar sempre com imagens quadradas para facilitar o uso das *quadtrees*. As seções 4.2 e 4.3 tratam do sistema de Fisher e a Seção 4.4 do esquema de Cardinal.

4.2 - Codificação Fractal de Imagens Usando Quadtrees

A etapa de codificação do esquema de Fisher se baseia no fluxograma de codificação por fidelidade almejada apresentado na Fig. 3.4 utilizando a métrica *rms*, pois esta escolha facilita a determinação dos valores de s_i e o_i. Neste esquema a coleção de blocos-imagem é gerada a partir do sistema de partição *quadtree*, e a coleção de domínios (conjunto domínio $\boldsymbol{D}$) é formada por blocos quadrados cujas dimensões são duas vezes as dimensões do bloco-imagem da imagem.

4.2.1 - Determinação dos blocos-imagem

A partição *quadtree* é a representação da imagem como uma árvore na qual cada nó (que corresponde a uma parte quadrada da imagem) possui quatro ramificações (que correspondem aos quatro quadrantes deste quadrado). A raiz da árvore é a imagem original, como mostrado no exemplo da Fig. 4.1.

Os blocos-imagem são selecionados da seguinte forma: particiona-se a imagem original em alguns níveis de *quadtree* (parâmetro do codificador: Nível mínimo da *quadtree*). Cada nó resultante será então comparado com todos os candidatos a bloco-domínio pertencentes ao Conjunto Domínio que tenham o dobro do tamanho do bloco-imagem. Cada candidato a bloco-domínio é filtrado e escalonados por um fator de 2 para ter o mesmo tamanho do bloco-imagem, calculando-se os parâmetros s_i e o_i de acordo com as expressões (4.2) mostradas mais adiante. Aplicam-se as transformações otimizadas a cada candidato a bloco-domínio e calculando-se a d_{rms} deste com o bloco-imagem correspondente.

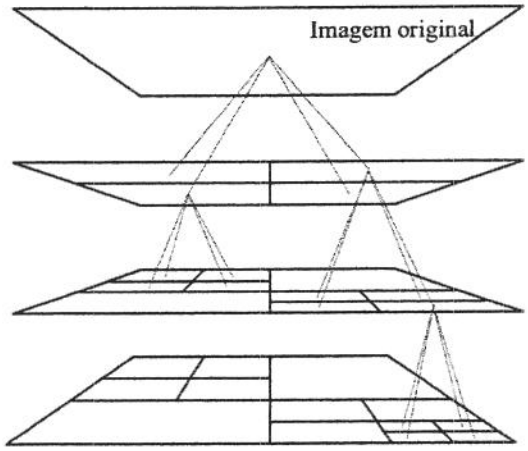


Fig. 4.1 – Exemplo de partição Quadtree

Compara-se a menor d_{rms} encontrada com a fidelidade mínima (parâmetro do codificador) e_c. Se a distância for maior que a fidelidade e o nível do quadtree for menor que o máximo permitido (parâmetro do codificador: Nível máximo da *quadtree*), particiona-se este bloco-imagem em quatro partes. Caso contrário escolhe-se como bloco-domínio e transformação os candidatos correspondentes e segue para codificar o próximo bloco-imagem.

4.2.2 - Determinação do Conjunto Domínio D

O esquema permite três diferentes tipos de conjunto domínio $\boldsymbol{D_1}$, $\boldsymbol{D_2}$ e $\boldsymbol{D_3}$. O primeiro contém um número aproximadamente igual de candidatos a domínios em cada tamanho de bloco. O segundo tem mais candidatos a domínios pequenos e menos candidatos a domínios grandes privilegiando a codificação dos detalhes das imagens. O terceiro tipo tem mais candidatos a domínios grandes e menos candidatos a domínios pequenos proporcionando, assim, uma melhor codificação dos blocos-imagem grandes, pois estes oferecem maior compressão (menor número de transformações a serem transmitidas/armazenadas).

Nos três esquemas, os candidatos a domínios são escolhidos como sendo blocos quadrados que tem o canto superior esquerdo alinhado com uma grade com passo *l*. Para cada tipo de conjunto domínio, o espaçamento da grade é escolhido em função do tamanho requerido do bloco-domínio como se segue:

D_1 : Espaçamento da grade igual a *l* em todos os tamanhos;

D_2 : Espaçamento da grade igual ao tamanho do bloco dividido por *l*. Este foi o esquema proposto por Jacquin [3];

D_3 : Inverso ao anterior, ou seja, o maior tamanho de bloco tem uma grade com espaçamento igual ao tamanho do menor bloco dividido por *l* e vice-versa.

4.2.3 - Cálculo do contraste e brilho

Para calcular os valores de s_i e o_i ótimos para cada candidato a bloco-domínio em uma dada orientação utiliza-se a expressão:

$$\min_{s_i\, o_i}\left(d_{rms}\left(\boldsymbol{r}_i,\ s_i \cdot \boldsymbol{d}_i + o_i \cdot \boldsymbol{1}\right)\right) \tag{4.1}$$

onde $\boldsymbol{r}_i$ é o bloco-imagem transformado em vetor, $\boldsymbol{d}_i$ é o bloco-domínio candidato já rotacionado/invertido e transformado em vetor e $\boldsymbol{1}$ é o vetor unitário. Neste ponto justifica-se a escolha da métrica *rms*, pois se utilizando esta métrica pode-se chegar à uma solução analítica para s_i e o_i na forma:

$$s_i = \frac{\sum \boldsymbol{d}_i \cdot \sum \boldsymbol{r}_i - N^2 \cdot \sum \boldsymbol{d}_i \boldsymbol{r}_i}{\left(\sum \boldsymbol{d}_i\right)^2 - N^2 \cdot \sum \boldsymbol{d}_i^2} \tag{4.2}$$

$$o_i = \frac{\sum \boldsymbol{d}_i \cdot \sum \boldsymbol{d}_i \boldsymbol{r}_i - \sum \boldsymbol{r}_i \cdot \sum \boldsymbol{d}_i^2}{\left(\sum \boldsymbol{d}_i\right)^2 - N^2 \cdot \sum \boldsymbol{d}_i^2} \tag{4.3}$$

onde:

$\sum \boldsymbol{r}_i$ = soma de todos os elementos do vetor $\boldsymbol{r}_i$;

$\sum \boldsymbol{d}_i$ = soma de todos os elementos do vetor $\boldsymbol{d}_i$;

$\sum \boldsymbol{r}_i \boldsymbol{d}_i$ = soma do produto elemento a elemento (produto interno) dos vetores $\boldsymbol{d}_i$ e $\boldsymbol{r}_i$;

$\sum \boldsymbol{d}_i^2$ = soma do quadrado de todos os elementos do vetor $\boldsymbol{d}_i$;

N^2 = número de elementos de cada vetor.

Note que os valores de s_i e o_i calculados por estas equações não são limitados e ainda serão quantizados posteriormente. Assim, para calcular o erro de aproximação é necessário utilizar-se os valores quantizados, pois serão estes os valores enviados ao decodificador. Outro fator de restrição

é o máximo valor de contraste permitido s_{max} (que é um parâmetro do codificador), se o valor calculado ultrapassar este máximo, então o valor máximo é usado para o cálculo do erro. Esta restrição é necessária para garantir convergência do mapa.

4.2.4 - Classificação dos domínios

Como observado anteriormente, se cada bloco-imagem tiver que ser comparado com todos os blocos-domínios de $\boldsymbol{D}$, a etapa de comparação domínio-imagem consumirá um tempo excessivo. Para evitar esta comparação, Fisher [7] propôs um esquema de classificação dos domínios de forma a reduzir o número de domínios comparados com cada bloco-imagem. Antes da codificação, todos os candidatos a domínio do conjunto domínio $\boldsymbol{D}$ são classificados. Durante o processo de codificação, o bloco-imagem também é classificado e apenas os domínios de classificação igual (ou próxima) são comparados e, assim, reduz-se significativamente o número de comparações domínio-imagem. A idéia de desenvolver esta classificação foi desenvolvida independentemente em [3] e [18].

Muitos esquemas de classificação são possíveis, mas o codificador proposto utilizou a classificação apresentada em [19]: o bloco quadrado é dividida em quatro quadrantes: esquerdo alto, direito alto, esquerdo baixo e direito baixo que são numerados seqüencialmente. Para cada quadrante são computados valores proporcionais à média e à variância dos *pixels*. Denotando os *pixels* do quadrante i por $r_1^i, ..., r_n^i$, para $i = 1, 2, 3, 4$, calcula-se:

$$A_i = \sum_{j=1}^{n} r_j^i \tag{4.3}$$

e posteriormente:

$$V_i = \sum_{j=1}^{n} \left(r_j^i\right)^2 - A_i^2 . \tag{4.4}$$

É sempre possível com uma isometria (combinação de rotações e inversões) orientar a sub-imagem de forma a ordenar os A_i de forma a classificá-la em uma das três superclasses (ilustradas na Fig. 4.2):

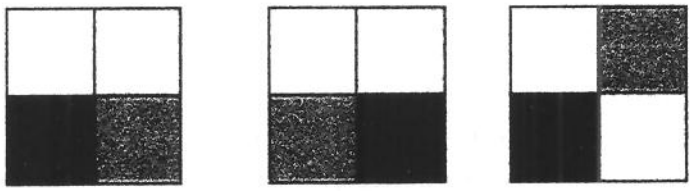
Superclasse 1) $A_1 \geq A_2 \geq A_3 \geq A_4$

Superclasse 2) $A_1 \geq A_2 \geq A_4 \geq A_3$

Superclasse 3) $A_1 \geq A_3 \geq A_2 \geq A_4$

Uma vez fixada a orientação, cada Superclasse tem 24 subclasses consistindo nas 24 possíveis ordenações dos valores de V_i, o que resulta em um total de 72 classes. Nesta classificação, se o valor cálculado de s_i for negativo, os ordenamentos nas classes acima são rearranjados.

Portanto cada domínio é classificado em duas orientações, uma para s_i positivo e uma para s_i negativo.



Superclasse 1 Superclasse 2 Superclasse 3

Fig. 4.2 – Superclasses de classificação por brilho

É interessante notar que, como se utiliza uma isometria para orientar o bloco durante a classificação de média, quando se compara um bloco-imagem e um bloco-domínio pode-se, através da comparação destas orientações, determinar a isometria que será usada no bloco domínio para que ele fique na mesma posição do bloco-imagem. Com isso, elimina-se a necessidade de comparar os blocos em cada uma das 8 orientações possíveis.

Este esquema exige como os parâmetros de codificador o limiar de tolerância *rms* ou fidelidade mínima (e_c), os níveis máximo e mínimo da *quadtree*, o tipo de conjunto domínio, o máximo fator de brilho permitido (s_{max}), e o número de bits usado para quantizar os valores s_i e o_i.

Assim, a imagem codificada é representada pela partição *quadtree* final da imagem indicando o particionamento dos blocos-imagem e, para cada bloco-imagem, conjunto (s_i , o_i , T, $\hat{d}_i$) que representa o melhor casamento. Onde s_i é o ajuste de contraste, o_i é o ajuste de brilho, T é a isometria utilizada e $\hat{d}_i$ é o bloco-domínio equivalente.

4.3 - Etapas de Armazenamento e Decodificação

4.3.1 - Armazenamento

Para aumentar a compressão, foi utilizado o seguinte sistema de alocação de bits:

Quadtree: Um bit para cada nível do *quadtree* para indicar a subdivisão do nível atual ou o envio da informação de transformação. No último nível de codificação não é enviado nenhum bit, pois não pode haver mais subdivisões.

Escala e deslocamento: Foram usado 5 bits para armazenar o fator de escala s_i e 7 bits para o deslocamento o_i. Não foi utilizada nenhuma codificação adaptativa para estes coeficientes embora suas distribuições não fossem uniformemente distribuídas. Ao invés disso optou-se pela avaliação da entropia de primeira ordem para a estimação do ganho de compressão.

Domínios: Os domínios são indexados e referenciados pelo seu índice. No entanto quando o fator

de escala é nulo, não são armazenados nem o domínio nem a orientação.

Orientação: Foram usados 3 bits para armazena-la pois são 8 orientações possíveis.

4.3.2 - Decodificação

O processo de decodificação consiste basicamente de realizar as iterações de W a partir de qualquer imagem inicial. Cada iteração de W é realizada como a seguir:

1) A informação da partição quadtree é usada para gerar todos os blocos-imagem;
2) Para cada bloco-imagem R_i, o bloco-domínio D_i correspondente é sub-amostrado por um fator de 2 realizando a média de cada grupo de 2 x 2 pixels não-sobrepostos;
3) Os pixels do domínio sub-amostrado são multiplicados pelo fator s_i, somados ao fator o_i e posicionado no bloco-imagem correspondente com a orientação indicada no código.

Estas iterações são repetidas até que a imagem final seja obtida, ou seja, até que o erro entre as imagens antes da iteração e depois da iteração seja pequeno o suficiente (menor que um limiar escolhido).

Fisher ainda propõe alguns métodos alternativos de realizar a decodificação, como a inversão de um matriz feita a partir dos coeficientes das transformações ou a utilização de uma aproximação de tamanho menor construída a partir da própria informação codificada, mas estes métodos não foram utilizados nesta tese.

Uma característica interessante da codificação fractal é a independência da resolução. Como os fractais apresentam detalhes em todas as escalas, uma vez codificada uma imagem por fractal, o processo de decodificação pode ser realizado em qualquer resolução. É claro que se a resolução usada na decodificação for maior que a resolução original, os detalhes gerados pelo processo iterativo podem não ser exatamente iguais aos reais, mas eles são auto-similares à imagem original.

4.4 - Codificador Acelerado de Cardinal

Cardinal [4] propôs um algoritmo acelerado para compressão fractal de imagens baseado na técnica de vetor de características elaborada originalmente por Saupe [20]. A idéia central do algoritmo é reduzir o processo de procura pelo casamento entre bloco-imagem e bloco-domínio ao problema geométrico clássico de busca pelos vizinhos mais próximos num espaço euclidiano.

Para este fim, a proposta de Saupe foi o cálculo de um vetor de características que representasse o bloco em um espaço $\mathcal{R}^n$, onde n é o número de pontos do bloco. Este vetor de característica é formado pelo operador projeção normalizada de Saupe, λ [20, Apêndice], dado por:

$$\lambda(x) = \frac{x - \langle x, e \rangle \cdot e}{\| x - \langle x, e \rangle \cdot e \|} \tag{4.5}$$

onde x é o bloco original transformado em vetor, $e = (1,...,1)/\sqrt{n}$ e n é o número de *pixels* existentes no bloco. $\lambda(x)$ é o vetor de características ou chave (*key*) do bloco. Deve-se notar que, aplicar o operador de Saupe, equivale a zerar a média do vetor e normalizá-lo.

O algoritmo proposto por Cardinal calcula uma chave para todos os blocos-domínio e todos os possíveis blocos-imagem da imagem a ser codificada. O cálculo da chave para os blocos domínio (chaves-domínio) são calculados após a filtragem de média e sub-amostragem pelo fator de 2 para que ele tenha o mesmo tamanho do bloco-imagem ao qual será comparado.

Assim, para cada nível de *quadtree* possível cria-se um espaço vetorial que armazena todas as chaves-domínio e chaves-imagem (chave para blocos-imagem). Neste ponto a procura pelo melhor casamento se restringiria à procura bloco-domínio mais próximo, mas como existe a restrição da contratividade ($s_i < 1$) do mapeamento, torna-se necessário a busca pelos blocos-domínio contidos na vizinhança próxima do bloco-imagem a ser codificado.

Para encontrar esta vizinhança, particiona-se recursivamente o espaço contendo as chaves de todos os blocos-imagem e domínio até que se atinja um limite pré-estabelecido μ de chaves-domínio em todos os sub-espaços resultantes. Este limite μ é um parâmetro do codificador.

Quando esta partição é completada, cada bloco-imagem é comparado apenas aos blocos domínios que possuem chaves dentro do mesmo subespaço que sua chave-imagem. Desta forma, reduz-se o número de comparações, acelerando o processo de procura pelo melhor casamento. Deve-se notar que a informação de particionamento deste espaço não precisa ser enviada, basta que se envie o índice do bloco-domínio escolhido.

A cada iteração do particionamento recursivo, o espaço de procura n-dimensional é dividido em duas partes por um hiperplano $(n-1)$-dimensional. A cada passo, o hiperplano divisor é representado por um vetor normalizado $v \in \mathcal{R}^n$ e por um valor escalar $\mu \in \mathcal{R}$, onde o hiperplano é ortogonal a v e contém o ponto $\mu \cdot v$, como pode ser observado na Fig. 4.3. A questão se concentra então em encontrar μ e v a cada passo para encontrar o hiperplano divisor correspondente.

Dado um plano divisor, para determinar se uma dada chave p_i está em um lado do hiperplano ou no outro, basta calcular a projeção $\langle p, v \rangle$ e verificar se esta projeção é maior ou menor que μ. Na Fig. 4.3, por exemplo, como $\langle p_1, v \rangle > \mu$ a chave p_1 está no lado A do hiperplano.

A complexidade do processo de particionamento é da ordem de $O(\, n(\, N_R + N_D\,)\,)$ para cada

nível de partição, onde N_R e N_D são, respectivamente, o número de blocos-imagem e blocos-domínio dentro do subespaço que vai ser particionado.

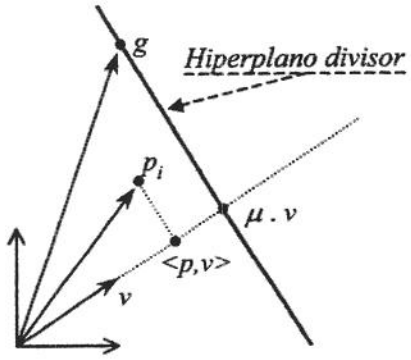


Fig. 4.3 – Ilustração do processo de particionamento

A solução ideal para realizar o particionamento do espaço foi apresentada em [21] e consiste em assumir v como o principal auto-vetor (denominado v_{opt}) da matriz de covariância da distribuição das chaves e μ como o auto-valor associado a v. Essa solução foi considerada ótima por Cardinal pois maximiza a variância residual das chaves e, portanto, maximiza a informação em cada lado do hiperplano divisor.

No entanto, como o cálculo da matriz de covariância requer um grande esforço computacional, Cardinal propôs um algoritmo rápido para encontrar uma aproximação deste hiperplano ótimo sem calcular explicitamente v_{opt}.

Este algoritmo consiste em encontrar o centro de gravidade das chaves-imagem com o objetivo de centrar a distribuição e, posteriormente, calcular a média das direções observadas a partir deste ponto até as chaves-imagem. Ao utilizar somente as chaves-imagem para encontrar os hiperplanos divisores, o algoritmo acelera muito o processo, uma vez que o número de blocos-imagem é muito menor do que o número de blocos-domínio. Essa aproximação pressupõe que as distribuições de chaves-imagem e chaves-domínio sejam parecidas, o que é razoável se aceitarmos a existência da auto-similaridade nas imagens.

O algoritmo proposto por Cardinal é formalmente descrito pelos seguintes passos:

1. Calcula-se $\{r_i \in \Re^n \mid i = 1,...,N_R\}$ e $\{d_i \in R^n \mid i = 1,...,N_D\}$, que são o conjunto de chaves-imagem e o conjunto de chaves-domínio, respectivamente.
2. Calcula-se o centro de gravidade g da distribuição das chaves-imagem:

$$g = (1/N_R)\sum_{i=1}^{N_R} r_i \tag{4.6}$$

3. Para cada $i \in 1,....,N_R$, calcule a projeção de cada r_i na esfera unitária centrada em g:

$$t_i = \frac{(r_i - g)}{\| r_i - g \|} \tag{4.7}$$

4. Calcula-se:

$$w = (1/N_R) \sum_{i=1}^{N_R} t_i \tag{4.8}$$

5. O hiperplano divisor procurado será ortogonal à direção dada por $v = w / \|w\|$ e conterá o ponto g. O valor correspondente a μ é dado pela projeção $\langle g, v \rangle$.

Uma vez que o hiperplano divisor tenha dividido o espaço de procura em subespaços pequenos o suficiente, é iniciado o processo de busca pelo melhor casamento de forma semelhante ao codificador de Fisher. Assim, dentro de cada subespaço, cada bloco-imagem será comparado com os blocos-domínio componentes desse mesmo subespaço. Cardinal não usou as transformações geométricas (isometrias) fractais e, portanto, para cada comparação bloco-imagem com bloco-domínio é feito um único cálculo de S_i e O_i (brilho e contraste). A comparação que gere menor RMS fornecerá o bloco-domínio equivalente, semelhantemente a Fisher.

Os parâmetros de codificação e a decodificação são basicamente iguais aos de Fisher, com duas exceções: a existência do novo parâmetro de codificação Número Máximo de Chaves-domain por subespaço e da ausência das transformações isométricas.

Cardinal comparou sua proposta com 3 métodos [17]:

1. Método randômico: no qual v é determinado segundo um particionamento qualquer;
2. Método ótimo: determinado pelo cálculo do v_{opt};
3. Método *k-d-tree* de partição.

Os resultados de Cardinal publicados em [17] validaram seu codificador como tendo desempenho superior aos demais métodos: proporcionou um aumento significativo de velocidade para a mesma qualidade de imagem.

4.5 - Conclusões

Nesta seção foram apresentados os codificadores fractais propostos por Fisher [7] e por Cardinal [17]. Estes codificadores foram usados como referência de comparação para os resultados obtidos na tese, pois o primeiro é uma base de referência para muitos artigos da literatura recente

em codificação fractal de imagens, e o segundo foi indicado pela *IEEE Transactions on Image Processing* para o artigo publicado sobre esta pesquisa.

Nos próximos capítulos serão abordadas as bases matemáticas da Transformada *Wavelet* Discreta e suas características, culminando com a apresentação do algoritmo SPIHT (*Set Partitioned in Hierarchical Trees*) que também foi utilizado como base de comparação para os resultados apresentados.

Capítulo 5

Transformada *Wavelet* Discreta

A expansão vertiginosa das aplicações de imagens digitais dos últimos anos tem despertado o interesse de pesquisadores das mais diversas áreas no intuito de desenvolver tecnologias e novas transformadas que solucionem os problemas da compressão de imagens. A transformada *wavelet* (TW) é fruto das contribuições destes pesquisadores concentrados especialmente nas áreas da matemática, física, estatística, computação gráfica e engenharia.

São muitas as vantagens das transformadas wavelet. Pode-se destacar a existência de algoritmos que permitem rápida implementação, excelente capacidade de concentrar a energia em poucos coeficientes e ausência de artefatos de blocagem.

Hoje, padrões importantes, tais como o JPEG 2000, MPEG-4 e MPEG-7, para compressão de imagem e vídeo fazem uso da transformada *wavelet*.

O objetivo deste capítulo é apresentar a transformada *wavelet* contínua 1D e 2D, sua interpretação através do conceito de banco de filtros e a expansão em série da *wavelet* (SW) uni e bidimensional. Este estudo se restringirá a funções quadraticamente integráveis 1D e 2D. O espaço de funções mensuráveis (quadraticamente integráveis) sobre o eixo real é definido como $L^2(\Re)$.

5.1 - Introdução: Waves x Wavelets

Matemáticos e engenheiros têm explorado transformações cujas funções-base têm duração limitada. Essas funções-base (chamadas de *wavelets* ou ondeletas) são ondas que variam em posição e freqüência. As transformadas baseadas nelas são chamadas de transformadas *wavelet* [22]. A Fig. 5.1 ilustra a diferença entre *Waves* (ondas) e *Wavelets* (ondeletas).

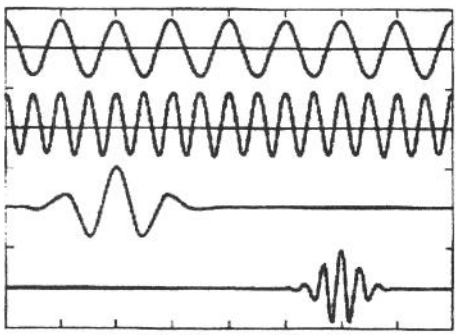


Fig. 5.1. Waves e Wavelets

A técnica conhecida por análise tempo-freqüência precedeu a TW, mas apresenta a mesma estrutura. Na análise tempo-freqüência o sinal é submetido a várias filtragens diferentes gerando, por exemplo, o gráfico da Fig. 5.2 que permite a análise do conteúdo do sinal. A TW que será apresentada a seguir também torna possível este tipo de análise.

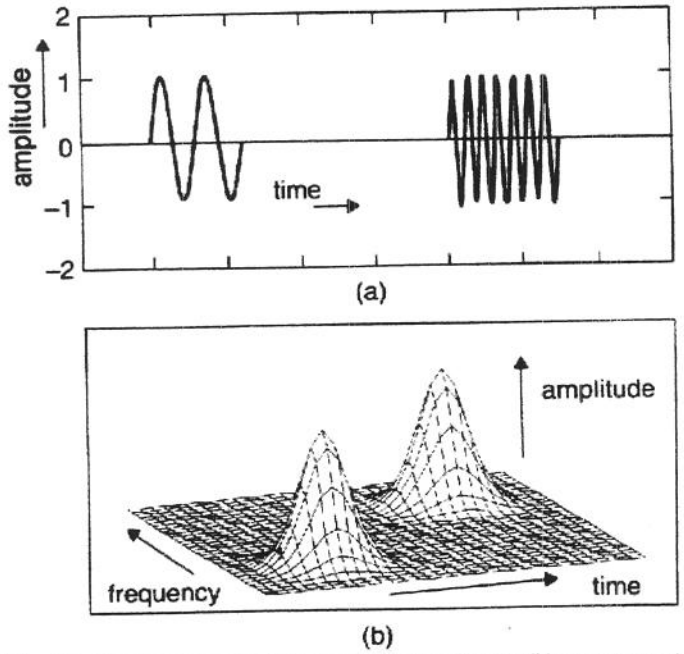


Fig. 5.2. Espaço tempo-freqüência: (a) sinal original; (b) representação

5.2 - Transformada *Wavelet* Contínua (TWC) Unidimensional

A TWC foi introduzida por Grossman e Morlet [23]. A função protótipo $\psi(x)$ é chamada de *wavelet*-básica ou *wavelet*-mãe, função oscilatória usualmente centrada na origem e que cai rapidamente a zero quando $|x| \to \infty$. Assim, $\psi(x) \in L^2(\Re)$.

As funções-base $\psi_{a,b}(x)$ são comumente chamadas de *wavelets*-filhotes, versões deslocadas e dilatadas ou comprimidas de $\psi(x)$ e capazes de gerar todo o espaço $L^2(\Re)$. Assim, sendo $a>0$ e b números reais:

$$\psi_{a,b}(x) = \frac{1}{\sqrt{a}} \psi\left(\frac{x-b}{a}\right) \tag{5.1}$$

Em (5.1), a é o parâmetro responsável pela dilatação ou compressão de $\psi(x)$, enquanto b é o parâmetro responsável pelo deslocamento de $\psi(x)$ ao longo do eixo x. A constante $a^{-1/2}$ em (5.1) é

o termo de normalização da energia da função com relação ao parâmetro a, assegurando que as normas das funções-base sejam iguais, pois:

$$\left\| f\left(\frac{x-b}{a}\right) \right\| = \sqrt{\int_{-\infty}^{\infty} \left| f\left(\frac{x-b}{a}\right) \right|^2 dx} = \sqrt{a}\,\|f(x)\| \tag{5.2}$$

Na Fig. 5.3 é apresentada a *wavelet* de Morlet para diferentes valores de a e b.

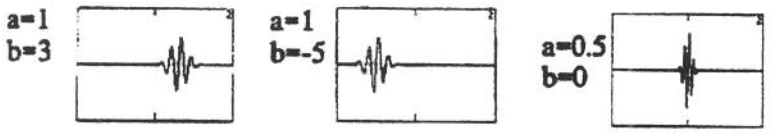


Fig. 5.3. Wavelets de Morlet.

Assim, seja a *wavelet* $\psi(x)$ uma função oscilatória e de curta duração, a TWC é uma transformada real e é definida pela seguinte expressão [22]:

$$W(a,b) \overset{\Delta}{=} \int_{-\infty}^{\infty} f(x) \cdot \psi_{a,b}(x)\,dx = < f, \psi_{a,b} > \tag{5.3}$$

onde $\psi_{a,b}(x)$ é dado em (5.1).

Para se obter a TWC inversa, é necessário que o seguinte critério de admissibilidade seja satisfeito [24] [25]:

$$C_\psi = \int_{-\infty}^{\infty} \frac{|\Psi(s)|^2}{|s|} ds < \infty \tag{5.4}$$

onde $\Psi(s)$ é o espectro da *wavelet*-básica real $\psi(x)$.

O critério de admissibilidade restringe a classe de funções que podem ser utilizadas como *wavelets*-básicas. Como s é denominador da integral, é necessário que:

$$\Psi(0) = 0 = \int_{-\infty}^{\infty} \psi(x)\,dx = 0 \tag{5.5}$$

Como $\Psi(\infty)=0$, pode-se ver que o espectro de amplitude da *wavelet-básica* admissível é similar à função de transferência de um filtro passa-faixas.

Se a constante C_ψ na equação (5.4) for finita, então existe a TWC inversa definida por:

$$f(x) = \frac{1}{C_\psi} \int_0^{\infty} \int_{-\infty}^{\infty} W(a,b)\,\psi_{a,b}(x)\,db\,\frac{da}{a^2} \tag{5.6}$$

5.3 - A TWC interpretada por Banco de Filtros

A TWC pode ser interpretada através do conceito de banco de filtros [22]. Definindo a *wavelet* escalonada de a e normalizada por $a^{-1/2}$:

$$\psi_a(x) = \frac{1}{\sqrt{a}}\psi\left(\frac{x}{a}\right) \tag{5.7}$$

Definindo também seu complexo conjugado refletido:

$$\tilde{\psi}_a(x) = \psi_a^*(-x) = \frac{1}{\sqrt{a}}\psi^*\left(\frac{-x}{a}\right) \tag{5.8}$$

Pode-se reescrever (5.3) e (5.6), respectivamente como:

$$W(a,b) = \int_{-\infty}^{+\infty} f(x)\psi_a(b-x)dx = f * \tilde{\psi}_a \tag{5.9}$$

$$f(x) = \frac{1}{C_\psi}\int_0^\infty \int_{-\infty}^{\infty}[f * \tilde{\psi}_a](b)\psi_a(b-x)db\frac{da}{a^2} = \frac{1}{C_\psi}\int_0^\infty [f * \tilde{\psi}_a * \psi_a](x)\frac{da}{a^2} \tag{5.10}$$

Nota-se que para *a* fixo, *W(a,b)* é a convolução de *f(x)* com o complexo conjugado refletido da *wavelet*-básica escalonada definida em (5.7).

A TWC pode ser interpretada como um banco de filtros lineares atuando sobre *f(x)*, conforme ilustrado na Fig. 5.4. Todas as saídas dos filtros juntas juntas compõem a TWC.

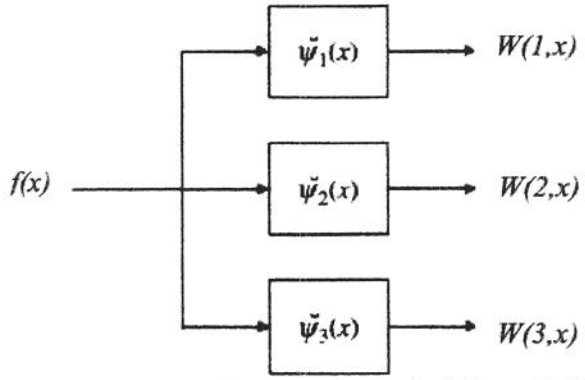


Fig. 5.4. TWC interpretada através do conceito de banco de Filtros

A TW permite uma visão simultânea dos resultados da transformação para diversos valores de escala, entretanto não envolve explicitamente o conceito de freqüência. Apesar da relação implícita existente entre escala e freqüência, esses conceitos são diferentes. Contudo, serão feitas analogias entre escala e freqüência de forma a elucidar determinados aspectos.

Alguns aspectos importantes devem ser observados. Primeiramente o fato de que há para o esquema da Fig. 5.4 uma única *wavelet*-básica. Além disso, $\tilde{\psi}_1(x)$, $\tilde{\psi}_2(x)$, $\tilde{\psi}_3(x)$, $\tilde{\psi}_4(x)$ são cada uma, a *wavelet*-básica escalonada com uma "freqüência" diferente e portanto cada uma é uma família de *wavelets* graças ao deslocamento *b*. Cada saída *W(a,b)* é um conjunto de coeficientes de mesma freqüência, correspondendo a uma linha do gráfico da Fig. 5.2.b.

Como em (5.8), se $\psi(x)$ for real e par (que é usualmente o caso) a reflexão e o conjugado não têm efeito. Logo, percebe-se que $\tilde{\psi}_a(x) = \psi_a(x)$.

De (5.10) percebe-se que as saídas dos filtros, cada qual filtrada novamente por $\psi_a(x)$ e corretamente escalonada, se combinam para reconstruir $f(x)$.

Como:

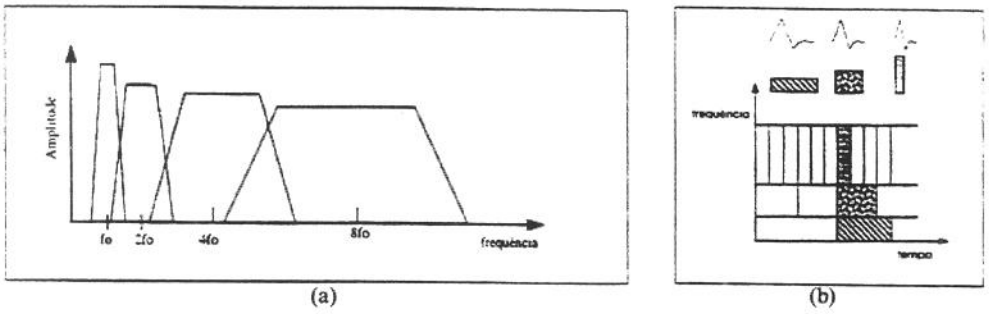
$$TF\{f(ax)\} = \frac{1}{|a|} F\left(\frac{s}{a}\right) \tag{5.11}$$

tem-se:

$$\Psi_a(s) = TF\{\psi_a(x)\} = \sqrt{a}\,\Psi(as) \tag{5.12}$$

A TWC possui janela de comprimento variável, ou seja, resoluções no tempo e freqüência variáveis. Seus filtros passa-faixas apresentam larguras simultaneamente diferentes definidas pelos valores de a.

Altos valores de a implicam na dilatação da *wavelet* definida em (5.7) no tempo (conseqüentemente contração em freqüência). Como se tratam de filtros passa-faixas, a contração em freqüência também desloca o centro da banda passante para valores mais baixos em freqüência. Portanto, aumentar a largura da *wavelet* no tempo, implica em analisar as baixas freqüências do sinal através de filtro estreito, como pode ser observado na Fig. 5.5.a. Analogamente, reduzindo a largura da *wavelet* no tempo, pode-se realizar a análise das altas freqüências do sinal através de filtros mais largos.



(a) (b)

Fig. 5.5. TWC: (a) Bandas de freqüências das janelas; (b) Resolução no plano tempo-freqüência

Desta forma, a resolução do sinal de saída de cada filtro é dependente da largura da *wavelet* correspondente a ele. Quanto maior a largura da *wavelet* no tempo, maior o número de componentes do sinal original usado na formação de cada coeficiente transformado a cada passo da convolução. Nesse caso há uma menor resolução no tempo e maior resolução em freqüência. Analogamente,

wavelets mais estreitas resultam em uma maior resolução no tempo e menor resolução em freqüência.

A TWC permite, portanto, a análise do sinal sob diferentes resoluções simultâneas. Esta característica faz com que alguns processamentos possam ser aplicados somente sobre determinados componentes do sinal como, por exemplo, a aplicação de um ganho sobre as camadas de detalhamento do sinal [26]. Esta é a grande vantagem da transformada *wavelet* sobre as demais transformadas.

5.4 - TWC Bidimensional

A TWC, $W(a,b)$, de uma função $f(x)$ é uma função de duas variáveis, logo "sobrecompleta". Dessa forma, representa um aumento considerável no volume de informação. Para funções de mais que uma variável, a TWC também aumenta em um a dimensionalidade.

Seja $\psi(x,y)$ uma função oscilatória e de curta duração. $\psi_{a;b_x,b_y}(x,y)$ são as *wavelets*-filhotes bidimensionais geradas a partir da *wavelet*-básica bidimensional $\psi(x,y)$ por:

$$\psi_{a;b_x,b_y}(x,y)=\frac{1}{a}\psi\left(\frac{x-b_x}{a},\frac{y-b_y}{a}\right) \tag{5.13}$$

onde b_x e b_y especificam a translação em duas dimensões.

A TWC-2D de $f(x,y)$ é dada por [22]:

$$W(a;b_x,b_y)=\left\langle f(x,y),\psi_{a;b_x,b_y}(x,y)\right\rangle=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}f(x,y)\psi_{a;b_x,b_y}(x,y)\,dxdy \tag{5.14}$$

Seja $\Psi(s_x,s_y)$ a transformada de Fourier 2D de $\psi(x,y)$, para obter a TWC 2D inversa é necessário que a seguinte condição de admissibilidade seja satisfeita [26]:

$$C_\psi=\int_0^\infty\int_0^{2\pi}\frac{|\Psi(r\cos\theta,rsin\theta)|^2}{r}\,dr\,d\theta<\infty \tag{5.15}$$

Caso $\psi(x,y)$ seja esfericamente simétrica, sua transformada de Fourier também é esfericamente simétrica e a condição de admissibilidade é simplificada na seguinte forma [25]:

$$C_\psi=\int_0^\infty\frac{|\Psi(a^{-1}s_x,a^{-1}s_y)|^2}{a}da<\infty \qquad ,\forall(s_x,s_y)\in R^2 \tag{5.16}$$

Se a constante C_ψ for finita, então existe a TWC 2D inversa, definida por:

$$f(x,y)=\frac{1}{C_\psi}\int_0^\infty\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}W(a;b_x,b_y)\psi_{a;b_x,b_y}(x,y)db_xdb_y\frac{da}{a^3} \tag{5.17}$$

Deve-se notar que, em virtude da redundância (visto que se trata de uma transformada

sobrecompleta), o valor potencial da TWC reside em processos de decomposição e análise e não em compressão de dados. Uma das aplicações poderia ser, por exemplo, realizar um determinado processamento somente sobre os componentes de alta freqüência.

5.5 - Expansão em Série da Wavelet (SW)

Através da discretização dos parâmetros de escala (a) e deslocamento (b) da transformada *wavelet* contínua pode-se obter a expansão em série *wavelet* (SW). A SW é a representação do sinal através da soma ponderada de escalonamentos e deslocamentos da *wavelet*-mãe. Logo, a SW de um sinal é um conjunto enumerável de coeficientes, que correspondem ao produto interno entre o sinal original e as *wavelets*-filhotes referentes aos pontos na grade bidimensional do domínio deslocamento-escala [26]. O parâmetro a sofre discretização exponencial, enquanto o parâmetro b sofre uma discretização proporcional a a, de forma que as suas expressões ficam:

$$a = a_0^j \tag{5.18}$$

$$b = k \cdot b_0 \cdot a_0^j = n \cdot a_0^j \tag{5.19}$$

Onde as constantes a_0 e b_0 são os comprimentos dos passos discretos de escala e deslocamento, respectivamente.

O parâmetro b sofre uma discretização proporcional ao parâmetro a para evitar que as *wavelets*-filhotes se sobreponham e apresentem redundância entre si. Quanto maior o valor de a, maior o número de componentes do sinal original para formar cada coeficiente transformado e, conseqüentemente, para evitar o excesso de redundância durante a convolução com $f(x)$, os passos de deslocamento da *wavelet* devem ser maiores. Analogamente, para valores pequenos de a, são necessários passos de deslocamento menores.

Substituindo as discretizações em (5.9), pode-se expressar os coeficientes $W(j,n)$ da SW de um sinal $f(x)$ por [24]:

$$W(j,n) = a_0^{-j/2} \int_{-\infty}^{\infty} f(x) \cdot \psi\left(a_0^{-j} \cdot x - n\right) dx \tag{5.20}$$

Para simplificar a notação, é comum referenciar-se $W(j,n)$ por $W_{j,n}$. Ao contrário da TWC, a SW só está definida para valores positivos de escala, o que não chega a ser uma restrição, uma vez que se pode utilizar uma *wavelet* refletida para cobrir as escalas negativas.

5.5.1 - ***Wavelets Diádicas***

O tamanho do passo da escala, a_0, é usualmente escolhido como sendo ½ de forma a facilitar a implementação, uma vez que dilatar um sinal por um fator de dois corresponde

simplesmente a tomar uma amostra sim e outra não do sinal. Assim, tem-se que:

$$\psi_{j,n}(x) = 2^{j/2} \cdot \psi(2^j \cdot x - n) \qquad j,n \in Z \tag{5.21}$$

Essas *wavelets* são chamadas *wavelets* diádicas porque, para a_0 = ½, as funções-base (*wavelets*-filhote) são formadas a partir da *wavelet*-básica através de escalonamentos binários (de 2^j) e deslocamentos diádicos (de $k/2^j$). Neste caso, o espaçamento da grade no plano deslocamento-escala fica como na Fig. 5.6.

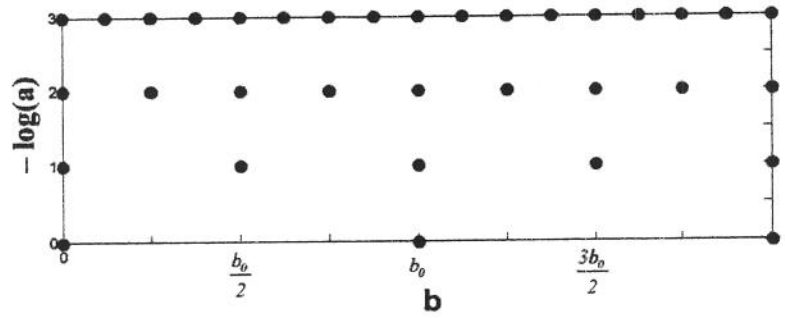


Fig. 5.6. Grade de amostragem no plano deslocamento-escala para wavelets diádicas (a_0=1/2)

Onde cada nó é um coeficiente-*wavelet* associado a uma *wavelet* $\psi_{j,n}(x)$. Caso seja escolhida uma *wavelet*-básica de norma unitária ($||\psi||=1$), as *wavelets*-filhotes $\psi_{j,n}(x)$ também apresentarão normas unitárias, o que implica na conservação de energia de $f(x)$ em $W_{j,n}$.

O processo de reconstrução do sinal $f(x)$ a partir dos seus coeficientes SW é semelhante ao processo de recuperação da TWC. No entanto, no caso da SW são necessários alguns cuidados. Por exemplo, imagine duas *wavelets* com escalas a_1 e a_2 respectivamente, sendo $a_1 > a_2$. A Fig. 5.7 mostra um esboço da TF de ambas.

Pode-se notar a partir da Fig. 5.7 que, se escolhermos $a_1 >> a_2$, pode-se obter grade muito esparsa verticalmente o que poderia resultar em $s_2 >> s_1$ e, portanto, a representação da SW iria desprezar uma parte da informação do sinal original, o que levaria a uma distorção irreversível do sinal, tornando impossível a reconstrução exata do sinal.

Por outro lado, uma grade muito densa verticalmente poderia até mesmo eventualmente indicar $s_1 > s_2$, o que à despeito de uma reconstrução fiel do sinal, acarretaria em redundância. Há, portanto, um compromisso entre fidelidade e eficiência de processamento na escolha da *wavelet*-básica e sua grade de operação. Em termos práticos, o ideal é que as grades sejam apenas densas o

suficiente para que a reconstrução seja realizada com um mínimo de perda e um máximo de eficiência. Tendo isso em mente, é usual buscar funções-base ortonormais.

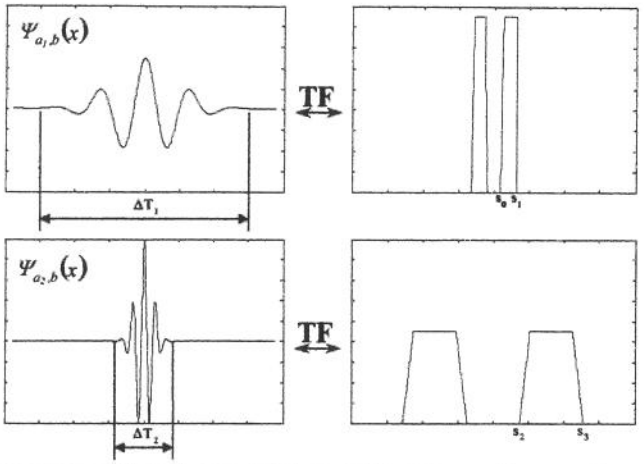


Fig. 5.7. Comparação entre as TF de duas wavelets com diferentes escalas *a*, sendo $a_1 > a_2$ e $\Delta T_1 > \Delta T_2$.

Assim, seja uma função $f(x) \in L^2(R)$ e a sua representação em série wavelet dada pelos coeficientes-*wavelet* $W_{j,n}$ expressos por:

$$W_{j,n} = \langle f, \psi_{j,n} \rangle \tag{5.22}$$

Se as funções-base $\psi_{j,n}(x)$, $-\infty < j,n < \infty$, formam uma base ortonormal de $L^2(R)$, ou seja:

$$\langle \psi_{j,k}, \psi_{l,m} \rangle = \delta_{j,k} \cdot \delta_{l,m} = \begin{cases} 1, & se\ j = l, k = m \\ 0, & cc \end{cases} \tag{5.23}$$

sendo l, m inteiros e $\delta_{j,n}$ sendo a função delta de Kronecker, então pode-se afirmar que o sinal $f(x)$ pode ser expresso de forma exata por: [24]:

$$f(x) = \sum_{j=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} W_{j,n} \cdot \psi_{j,n}(x) \tag{5.24}$$

Portanto, a série em (5.24) converge em $L^2(R)$:

$$\lim_{M_1,N_1,M_2,N_2 \to \infty} \left\| f(x) - \sum_{j=-M_2}^{N_2} \sum_{j=-M_1}^{N_1} W_{j,n} \cdot \psi_{j,n} \right\| = 0 \tag{5.25}$$

Assim, para as *wavelets* diádicas, os coeficientes da SW são dados por [24]:

$$W_{j,n} = \langle f, \psi_{j,n} \rangle = 2^{j/2} \int_{-\infty}^{\infty} f(x) \cdot \psi(2^j x - n) dx \tag{5.26}$$

Em (5.24) a função contínua é representada por uma dupla seqüência infinita. Contudo, conforme já dito, se $\psi(x)$ for escolhida apropriadamente de acordo com o sinal original, pode-se

truncar a série sem que haja grande erro de aproximação. Se *f(x)* tiver suporte compacto (i.e., se tiver duração finita) e se a *wavelet*-básica for bem localizada (i.e., decaia rapidamente a zero na medida em que dista da origem), então muitos dos coeficientes com altos valores de $|n|$ resultantes apresentarão pequenas amplitudes, podendo ser negligenciados dependendo da aplicação. Já para altos valores de $|j|$ a área das funções-base tendem a zero e portanto os coeficientes relacionados a ela podem ser negligenciados. Como conseqüência, o esforço computacional pode ser reduzido.

5.5.2 - *SW bidimensional*

A SW-2D é também uma extensão do caso unidimensional. Assim, fixando os valores de a_0 e b_0 faz-se $a = a_0^j$; $b_x = n_x \cdot a_0^j$ e $b_y = n_y \cdot a_0^j$. A SW-2D é, então, definida por:

$$W_{j;n_x,n_y} = a^{-j} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y) \cdot \psi\left(a_0^{-j} x - n_x, a_0^{-j} y - n_y\right) dx dy \tag{5.27}$$

e as *wavelets* diádicas ($a_0 = 1/2$, $b_0 = 1$) bidimensionais por:

$$\psi_{j;n_x,n_y}(x,y) = 2^j \psi\left(2^j x - n_x, 2^j y - n_y\right) \tag{5.28}$$

5.6 - Introdução aos Princípios da Transformada *Wavelet* Discreta

Antes de introduzir a Transformada Wavelet Discreta propriamente dita, será abordada brevemente uma técnica que levou ao desenvolvimento de sua implementação: a codificação por subbanda [22]. O objetivo desta seção é colocar o princípio de funcionamento da implementação da TWD para detalhá-la no próximo capítulo.

5.6.1 - Codificação e Decodificação por Subbanda

A codificação por subbanda é uma técnica que decompõe o sinal 1D ou 2D em componentes limitados em faixas de freqüência e os representa sem redundância e de forma a possibilitar a reconstrução do sinal sem erros [22] [27].

Seja o sinal *f(x)* limitado em banda e com transformada de Fourier ***F(s)***:

$$F(s) = 0 \quad para \quad |s| \geq s_N \tag{5.29}$$

A codificação por subbanda começa com a partição do eixo de freqüências em intervalos disjuntos de forma a permitir a implementação de filtros passa-faixas aos quais o sinal será submetido. Podem ser adotados M intervalos disjuntos de comprimento $2s_N/M$ produzindo M sinais de subbanda com S_N/M cada. Componentes de freqüências diferentes aparecerão em diferentes canais de subbanda.

Supondo dois canais de subbanda, o intervalo de freqüências é particionado no ponto $s_N/2$

submetendo o sinal original $f(i.\Delta x)$ de N pontos a dois filtros (usando convolução circular) que gerarão dois sinais de subbanda de $N/2$ pontos: um sinal meia-banda baixa e um sinal meia-banda alta. O sinal meia-banda baixa, $y_0(i.\Delta x)$, consiste de uma versão de baixa resolução ("borrada") de $f(i\Delta x)$. Já o sinal meia-banda alta, $y_1(i.\Delta x)$, consiste de uma versão que contém basicamente os detalhes de $f(i\Delta x)$

Seja $h(i.\Delta x)$ o filtro passa-baixas meia-banda, assim chamado por deixar passar somente a banda $[-s_N/2,\ s_N/2]$, ou seja, a metade da banda de freqüências do sinal. Seja também $g(i.\Delta x)$ o filtro passa-faixas meia-banda. Para gerar os dois sinais de subbanda $y_0(i.\Delta x)$ e $y_1(i.\Delta x)$, somente é necessária a filtragem de $f(i.\Delta x)$ com $h(i.\Delta x)$ e $g(i.\Delta x)$, seguidas pela subamostragem de cada saída. Assim,

$$y_0(2i\cdot\Delta x)=\sum_k f(k\cdot\Delta x)\cdot h((-k+2i)\Delta x) \tag{5.30}$$

$$y_1(2i\cdot\Delta x)=\sum_k f(k\cdot\Delta x)\cdot g((-k+2i)\Delta x) \tag{5.31}$$

Nesta técnica, $y_1(i.\Delta x)$ contém exatamente a informação de alta freqüência que foi eliminada de $f(i.\Delta x)$ durante a geração de $y_0(i.\Delta x)$. Assim, pode-se dizer que $y_1(i.\Delta x)$ e $y_0(i.\Delta x)$ contém juntos toda a informação presente no sinal original $f(i.\Delta x)$ [22]. Portanto, tem-se:

$$f(i\cdot\Delta x)=y_0(i\cdot\Delta x)+y_1(i\cdot\Delta x)=f(i\cdot\Delta x)*h(i\cdot\Delta x)+f(i\cdot\Delta x)*g(i\cdot\Delta x) \tag{5.32}$$

pois

$$H(s)+G(s)=1 \tag{5.33}$$

Assim, os sinais de subbanda $y_0(i.\Delta x)$ e $y_1(i.\Delta x)$ combinados contém toda a informação necessária para a recuperação do sinal original $f(i.\Delta x)$ sem erros no decodificador. Para a reconstrução do sinal original basta superamostrar esses dois sinais codificados em subbanda, interpolá-los com $2h(i.\Delta x)$e $2g(i.\Delta x)$, e em seguida, somá-los. Matematicamente:

$$f(i\cdot\Delta x)=2\sum_k\left[y_0(2k\cdot\Delta x)\cdot h((-i+2k)\Delta x)+y_1(2k\cdot\Delta x)\cdot g((-i+2k)\Delta x)\right] \tag{5.34}$$

O processo completo encontra-se esquematizado na Fig. 5.8.

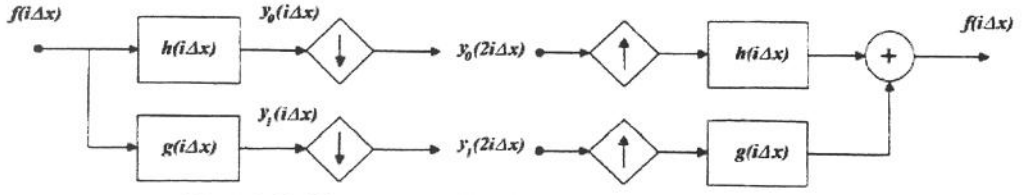


Fig. 5.8. Codificação por subbanda de duas bandas e reconstrução.

5.6.2 - Introdução ao Algoritmo Rápido para TWD

Mallat [28] definiu um algoritmo iterativo para implementar a transformada wavelet discreta de uma forma mais eficiente que calcular o conjunto completo de produtos internos. O algoritmo de Mallat é também conhecido como *Mallat's Herringbone Algorithm* ou FWT (*Fast Wavelet Transform*).

Nesse algoritmo, após aplicar a codificação por subbanda em dois canais ao sinal $f(i.\Delta x)$, o sinal meia-banda baixa $y_0(i.\Delta x)$ é novamente submetido à mesma codificação por subbanda em dois canais. Esse procedimento nos leva a um sinal meia-banda alta com $N/2$ coeficientes e dois sinais subbanda com $N/4$ coeficientes correspondendo ao primeiro e segundo quartos do intervalo $[0,s_N]$.

O processo iterativo é realizado até que seja obtido um único coeficiente conforme esquematizado na Fig. 5.9.a. Os coeficientes transformados resultantes dessa implementação são, portanto, o coeficiente passa-baixas e o conjunto de coeficientes meia-banda alta codificados; totalizando N coeficientes.

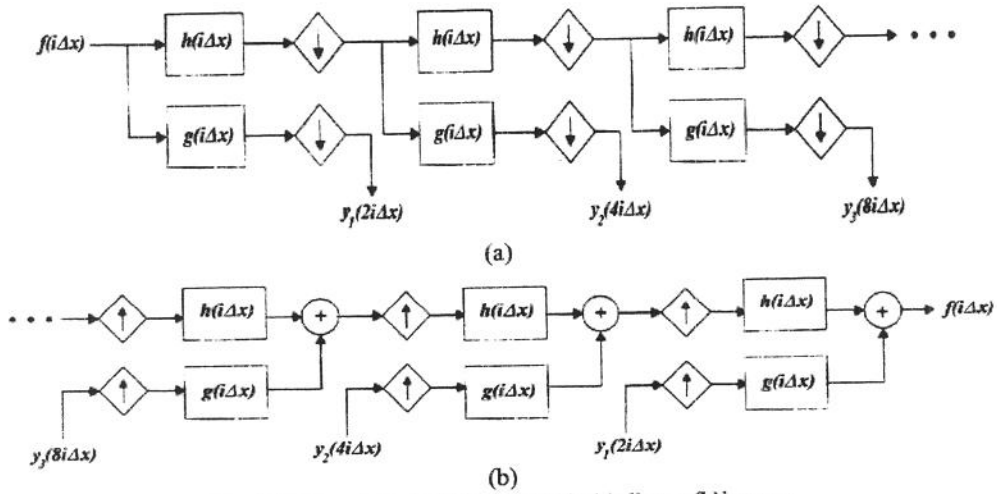


Fig. 5.9. Algoritmo de Mallat (TWD): (a) direto; (b)inverso.

Nota-se que cada conjunto de coeficientes transformados é obtido através da convolução de $f(i.\Delta x)$ repetidamente com $h(i.\Delta x)$ e somente uma vez com $g(i.\Delta x)$. Daí as funções-base da TWD, $\psi_{j,n}(x)$ serem a reflexão de $g(i.\Delta x)$ e de outras funções derivadas da convolução de $g(i.\Delta x)$ repetidamente com $h(i.\Delta x)$. O processo de reconstrução de $f(i.\Delta x)$ encontra-se esquematizado na Fig. 5.9.b.

5.7 - Conclusões

Como o objetivo deste trabalho é propor um esquema Híbrido Fractal-*Wavelet* para a compressão de imagens, neste capítulo foi apresentada uma introdução à transformada *wavelet* contínua 1D e 2D, a série da *wavelet* 1D e 2D, bem como suas características e princípios de funcionamento. Abordada a TWD e a introdução à implementação prática através do algoritmo rápido de Mallat, pode-se apresentar essa implementação de forma completa através da análise multiresolução no Capítulo 6.

A análise multiresolução permite determinar os filtros $h(i.\Delta x)$ e $g(i.\Delta x)$ de Mallat a partir das funções-base da TWD.

Capítulo 6

Análise Multi-Resolução e Projeto da TWD

A análise multi-resolução é uma técnica que permite analisar o sinal através da decomposição de seu conteúdo em um conjunto de detalhes em diferentes resoluções. O objetivo deste Capítulo é fornecer a base completa da implementação da TWD para sinais unidimensionais através da análise multi-resolução, gerando os sinais passa-baixa e de detalhes após a transformação *wavelet*.

6.1 – Aproximação por Multi-Resolução em $L^2(\Re)$: Propriedades

Seja um sinal $f(x) \in L^2(\Re)$ definido no intervalo $0< x <x_0$, então define-se como resolução de $f(x)$ a quantidade de pontos para os quais ele é definido dentro desse intervalo [26][28]. Por exemplo, uma função contínua $f(x)$ é conhecida para todos os pontos do intervalo e, portanto, tem resolução infinita. Considera-se uma *aproximação* de $f(x)$ original, o sinal resultante de uma alteração na resolução original.

Um sinal definido para r_j pontos do intervalo possui resolução r_j. Dada uma seqüência aproximações do sinal em resoluções crescentes $\{r_j , j \in Z\}$, pode-se definir como os *detalhes* do sinal em uma dada resolução r_j como sendo a diferença de informação entre a aproximação com resolução r_{j+1} e a aproximação com resolução r_j, mais baixa [28].

Seja a aproximação do sinal $f(x)$ com resolução 2^j denotada por $A_{2^j} \cdot f(x)$. O operador linear A_{2^j} é responsável pela projeção de $f(x)$, gerando sua aproximação com resolução 2^j. Dado que a aproximação de $f(x)$ é decorrente de uma mudança na resolução de $f(x)$, na prática pode-se entendê-la como o resultado de uma mudança na taxa de amostragem do sinal original e, dessa forma, encarar A_{2^j} como, por exemplo, um pente de Dirac com 2^j amostras no intervalo aberto $(0,x_0)$. O operador linear A_{2^j} apresenta as seguintes propriedades [28]:

Propriedade 1

Se $A_{2^j} \cdot f(x)$ é a aproximação da função $f(x)$ com resolução 2^j, então $A_{2^j} \cdot f(x)$ não é modificada se a operação for repetida no nível de resolução 2^j. Ou seja, $A_{2^j} \cdot A_{2^j} = A_{2^j}$. O operador A_{2^j} é um operador de projeção no subespaço vetorial $V_{2^j} \subset L^2(\Re)$. O subespaço vetorial V_{2^j} pode

ser interpretado como o conjunto de todas as possíveis aproximações de todas as funções $L^2(\Re)$ no nível de resolução 2^j.

O conjunto dos subespaços vetoriais V_{2^j} formam o espaço vetorial $\{V_{2^j}, j \in Z\}$ que será referenciado ao longo do texto como $\{V_{2^j}\}$ de forma a diferenciá-lo de cada V_{2^j} em particular. O termo $\{V_{2^j}\}$ se refere, portanto, ao espaço que contém todas as possíveis sub-amostragens das funções $f(x) \in L^2(\Re)$ em todas as resoluções 2^j, $j \in Z$.

Propriedade 2

Dentre todas as possíveis aproximações de $f(x)$ no nível de resolução 2^j, $A_{2^j} \cdot f(x)$ será designada como sendo a função que mais se aproxima de $f(x)$ de modo que:

$$\forall y(x) \in V_{2^j} \ , \ |y(x) - f(x)| \geq |A_{2^j} \cdot f(x) - f(x)| \tag{6.1}$$

Dessa forma, o operador A_{2^j} calcula a projeção ortogonal da função no espaço vetorial V_{2^j}.

Propriedade 3

A aproximação de $f(x)$ no nível de resolução 2^{j+1} contém toda a informação para calcular a aproximação de $f(x)$ no nível de resolução 2^j. Uma vez que A_{2^j} é um operador de projeção em V_{2^j}, tem-se que:

$$\forall j \in Z \ , \ V_{2^j} \subset V_{2^{j+1}} \tag{6.2}$$

Recursivamente, obtém-se a hierarquia de subespaços:

$$\ldots \subset V_{2^{-1}} \subset V_{2^0} \subset V_{2^1} \subset V_{2^2} \subset \ldots \tag{6.3}$$

Propriedade 4

A operação aproximação independe do nível de resolução e os subespaços das aproximações podem ser obtidos a partir de outros subespaços, através da dilatação ou compressão das aproximações:

$$\forall j \in Z \ , \ A_{2^j} \cdot f(x) \in V_{2^j} \Rightarrow A_{2^j} \cdot f\left(\frac{x}{2}\right) \in V_{2^{j+1}} \tag{6.4}$$

Propriedade 5

Sendo $f(x)$ definida no intervalo $0 \leq x \leq x_0$. Normalizando esse intervalo por x_0, a aproximação $A_{2^j} \cdot f(x)$ pode ser caracterizada por 2^j amostras por unidade de comprimento.

Quando *f(x)* é deslocada por um valor proporcional a 2^j, $A_{2^j} \cdot f(x)$ é deslocada do mesmo valor, logo, o operador A_{2^j} é invariante com o deslocamento.

Propriedade 6

Sempre há perda de informação quando é feita uma aproximação de *f(x)*. Essa perda é inversamente proporcional à resolução. Assim:

$$\lim_{j \to +\infty} V_{2^j} = \bigcup_{j=-\infty}^{+\infty} V_{2^j} = L^2(R) \tag{6.5}$$

$$\lim_{j \to -\infty} V_{2^j} = \bigcap_{j=-\infty}^{+\infty} V_{2^j} = \{0\} \tag{6.6}$$

De (6.5) e (6.6), percebe-se que à medida que a resolução aumenta tendendo a $+\infty$, a aproximação converge para o sinal original. À medida que a resolução diminui, a aproximação passa a conter cada vez menos informação, convergindo para zero.

Definição 10.1: "Qualquer conjunto de subespaços vetoriais $\{V_{2^j}\}$ que satisfaçam as propriedades 1 a 6 é dito uma Aproximação por Multi-Resolução em $L^2(\Re)$. O conjunto de operadores $\{A_{2^j}\}$ associados a esse conjunto de espaços vetoriais gera as aproximações de qualquer função $f(x) \in L^2(\Re)$ em todos os níveis de resolução 2^j, para $j \in Z$".

Teorema 6.1: "Seja uma aproximação por multi-resolução em $L^2(\Re)$, $\{V_{2^j}, j \in Z\}$. Dado *j*, há um único operador A_{2^j} que faz a projeção ortogonal de *f(x)* na base ortonormal de V_{2^j}. Então, existe uma única função $\phi(x) \in L^2(\Re)$ responsável pela formação de todas as bases ortonormais de todos os subespaços. Essa função $\phi(x)$ é denominada função-escala que define a família [28]:

$$\phi_{j,n}(x) = 2^{j/2}\phi(2^j x - n) \qquad , \; n \in Z \tag{6.7}$$

Essa família, $\{\phi_{j,n}\}$, através da variação de *n*, forma uma base ortonormal para cada V_{2^j}. A relação entre $\{\phi_{j,n}\}$ e V_{2^j} é ilustrada na Fig. 6.1, que exemplifica o caso para $j \in \{-1,0,1\}$.

Note que as bases dos três subespaços vetoriais são diferentes, porém geradas a partir de escalonamentos e deslocamentos da função-escala $\phi(x)$. Na Fig. 6.1 somente está representada a base ortonormal de $V_{2^{-1}}$.

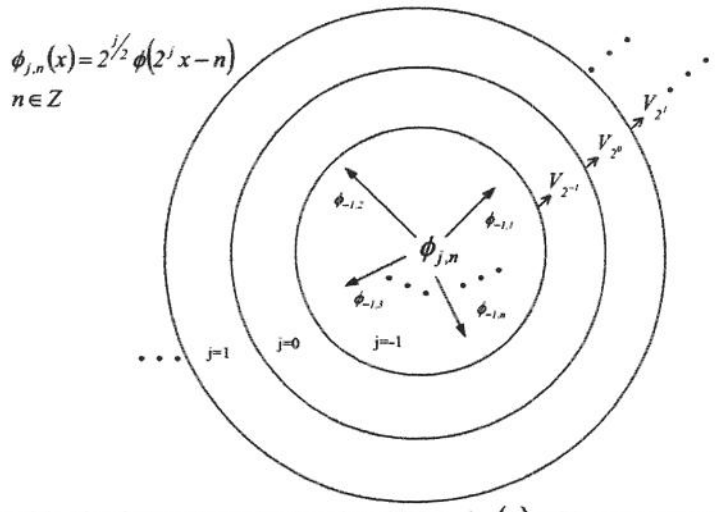


Fig. 6.1. Abordagem ilustrativa da relação entre $\phi_{j,n}(x)$ e V_{2^j} para $j \in [-1,1]$

O Teorema 6.1 mostra que a aproximação por multi-resolução $\{V_{2^j}, j \in Z\}$ é completamente caracterizada pela função-escala $\phi(x)$. A função escala deve ser continuamente diferenciável e o decaimento assintótico de $\phi(x)$ e de sua derivada $\phi'(x)$ no infinito deve satisfazer:

$$\phi(x)=O(x^{-2}) \qquad e \qquad |\phi'(x)|=O(x^{-2}) \tag{6.9}$$

Assim, a aproximação de $f(x)$ no subespaço vetorial V_{2^j} pode ser obtida através da decomposição de $f(x)$ na função na base ortonormal definida no Teorema 6.1:

$$\forall f(x)\in L^2(\Re)\ ,\ A_{2^j}\cdot f(x)=\sum_{n=-\infty}^{\infty}\langle f(u),\phi_{j,n}(u)\rangle\cdot\phi_{j,n}(x) \tag{6.10}$$

Sendo que os coeficientes dessa aproximação podem ser obtidos em função dos produtos internos entre o sinal $f(x)$ e a funções-base:

$$A^{d}_{2^j}\cdot f=\{\langle f(u),\phi_{j,n}(u)\rangle\ ,n\in Z\} \tag{6.11}$$

onde $A^d_{2^j}\cdot f$ é a aproximação discreta de $f(x)$ na resolução 2^j [28].

6.2 - Implementação da TWD via Multi-Resolução

Como a TWD é uma transformada discreta, para a implementação da TWD através da multi-resolução é necessária a adaptação da base matemática apresentada na seção anterior ao caso de sinais discretos.

Como os sinais originais possuem uma resolução finita, o mais usual é que se normalize a sua representação de forma que esse sinal seja referenciado como tendo uma resolução unitária, ou seja, 1 ponto por unidade de comprimento. Assim representa-se o sinal original como a aproximação do sinal no nível de resolução unitário $A_{2^j}^d \cdot f$.

Pelas propriedades da multi-resolução, todas as aproximações discretas $A_{2^j}^d \cdot f$ para $j<0$ podem ser calculadas a partir da aproximação $A_1^d \cdot f$. Nessa seção serão mostrados algoritmos simples e iterativos para o cálculo destas aproximações discretas [26][28].

Seja $\{V_{2^j}, j \in Z\}$ e $\phi(x)$, respectivamente, uma aproximação por multi-resolução e sua função-escala correspondente. De acordo com o Teorema 6.1, a família $\{\phi_{j,n}\}$ é uma base ortonormal para cada V_{2^j} [28][29]. Como $V_{2^j} \subset V_{2^{j+1}}$ (Propriedade 3), pode-se então expandir as funções $\{\phi_{j,n}\}$ na base ortonormal de $V_{2^{j+1}}$:

$$\phi_{j,n}(x) = \sum_{k=-\infty}^{\infty} \langle \phi_{j,n}(y), \phi_{j+1,k}(y) \rangle \phi_{j+1,k}(x) \tag{6.12}$$

Fazendo $u=2^{-j-1}y+n/2^j$ no produto interno de (6.12), tem-se:

$$\langle \phi_{j,n}, \phi_{j+1,k} \rangle = \langle 2^{j/2}\phi(2^j u - n),\ 2^{(j+1)/2}\phi(2^{j+1}u - k) \rangle =$$
$$2^{j/2+(j+1)/2} \int_{-\infty}^{\infty} \phi(2^{-1}y) \cdot \phi(y-(k-2n)) 2^{-j-1} dy = 2^{-1/2} \langle \phi(2^{-1}u), \phi(u-(k-2n)) \rangle \tag{6.13}$$

Assim, retornando à (6.12) tem-se:

$$\varphi_{j,n}(x) = \sum_{k=-\infty}^{\infty} \langle \varphi_{-1,0}(u), \varphi_{0,k-2n}(u) \rangle \varphi_{j+1,k}(x) \tag{6.14}$$

Aplicando o produto interno de $f(x)$ com ambos os lados de (6.14), tem-se:

$$\langle f, \phi_{j,n} \rangle = \sum_{k=-\infty}^{\infty} \langle \phi_{-1,0}(u),\ \phi_{0,k-2n}(u) \rangle \langle f(u), \phi_{j+1,k}(u) \rangle \tag{6.15}$$

A partir dessa equação (6.15) pode-se então calcular a expressão da aproximação do sinal no nível de resolução 2^j à partir da aproximação no nível de resolução imediatamente superior 2^{j+1}, como o indicado na propriedade 3.

Definindo o filtro discreto $h(m)$ de resposta impulsiva como:

$$h(m) = \langle \phi_{-1,0}(u),\ \phi_{0,m}(u) \rangle \quad \forall n \in Z \tag{6.16}$$

É possível encarar as equações (6.14) e (6.15) como sendo processos de filtragem. Assim pode-se reescrever essas equações na forma:

$$\phi_{j,n}(x) = \sum_{k=-\infty}^{\infty} h(k-2n)\cdot\phi_{j+1,k}(x) \tag{6.17}$$

$$\left\langle f, \phi_{j,n} \right\rangle = \sum_{k=-\infty}^{\infty} h(k-2n)\cdot\left\langle f, \phi_{j+1,k} \right\rangle \tag{6.18}$$

Ou seja, para cada n, obtém-se um coeficiente da projeção de $f(x)$ na resolução 2^j a partir de todos os coeficientes da projeção de $f(x)$ na resolução 2^{j+1}. Dessa forma, (6.18) mostra que $A^d_{2^j}\cdot f$ pode ser calculada convoluindo $A^d_{2^{j+1}}\cdot f$ com $h(-n)$ e dizimando o resultado por um fator de 2, conforme esquematizado no ramo superior da Fig. 46.2.

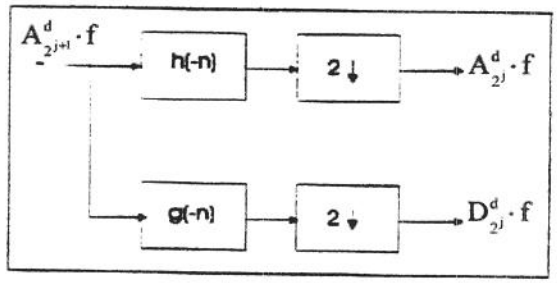


Fig. 6.2. Diagrama em blocos do esquema de decomposição.

Utilizando recursivamente esse sistema de filtros, pode-se calcular todas as aproximações $A^d_{2^j}\cdot f$ para $j<0$ à partir de $A^d_1\cdot f$. Esse processamento é conhecido como codificação piramidal [30]. Assim, supondo que a aproximação discreta de $f(x)$ no maior nível de resolução seja completamente caracterizada por N coeficientes: $A^d_1\cdot f = \alpha_n$ para $1\leq n\leq N$; cada aproximação discreta $A^d_{2^j}\cdot f$ para $j<0$ terá $2^j.N$ coeficientes.

Para se obter a relação entre a função-escala $\phi(x)$ e o filtro $h(n)$ basta fazer $j = n = 0$ na equação (6.17), aplicando em seguida a Transformada de Fourier:

$$\phi(x) = \sum_{k=-\infty}^{\infty} h(k)\cdot 2^{1/2}\phi(2x-k) \tag{6.19}$$

$$\phi(w) = \sum_{k=-\infty}^{\infty} h(k)\frac{\phi(0{,}5w)}{2^{1/2}}e^{-i\,0{,}5wk} = \frac{\phi(0{,}5w)}{2^{1/2}}H(0{,}5w) \tag{6.20}$$

$$\phi(w) = \phi(0)\cdot\prod_{m=1}^{\infty}\frac{H\left(w\cdot 2^{-m}\right)}{2^{1/2}} \tag{6.21}$$

Em [31] prova-se que para qualquer análise multi-resolução, $\phi(0)=1$. Isto implica que a família de funções $\{\phi_{j,n}\}$ não são *wavelets*, uma vez que não obedecem à condição de

admissibilidade (equação (5.4)). Em [32] demonstra-se que se $h(k)$ é um filtro FIR de comprimento N (não-nulo apenas para $0 \leq k \leq N-1$), então $\phi(x)$ é não-nula somente para $0 \leq x \leq N-1$. Essa propriedade permite que se determine uma função-escala $\phi(x)$ bem localizada em x através da imposição um conjunto de restrições a $h(k)$. Sendo $\boldsymbol{H(w)}$ a TF de $h(n)$:

$$\boldsymbol{H(w)} = \sum_{n=-\infty}^{\infty} h(n) e^{-inw} \tag{6.22}$$

$\boldsymbol{H(w)}$ deve satisfazer as propriedades (6.23 - 6.26) [28]:

$$H(0) = \sum_{k=-\infty}^{\infty} h(k) = \sqrt{2}\text{, obtida fazendo } w=0 \text{ em (6.21).} \tag{6.23}$$

$$h(n) = O\left(x^{-2}\right) \text{ quando } x \to \infty \tag{6.24}$$

$$|H(w)|^2 + |H(w+\pi)|^2 = 2\text{, } \forall w \text{ [14].} \tag{6.25}$$

$$|H(w)| \neq 0 \quad \text{para } w \in \left[0, \pi/2\right] \tag{6.26}$$

Pode-se notar que o espectro do filtro $h(n)$ é nulo para $w=\pi$, ou seja, ele é um filtro passa-baixas.

Com isso terminamos a expressão do cálculo da aproximação de resolução 2^j à partir da resolução 2^{j+1}. Na próxima seção definir-se-á o sinal de detalhes, ou seja, o sinal que contém a diferença de informação entre essas aproximações.

6.3 - O Sinal de Detalhamento

Seja W_{2^j} o subespaço vetorial que é o complemento ortogonal do subespaço vetorial V_{2^j}, de forma que:

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

$$V_{2^j} \oplus W_{2^j} = V_{2^{j+1}} \tag{6.27}$$

conforme exemplificado na Fig. 6.3.

O sinal formado a partir da diferença de informação entre as aproximações nos níveis de resolução 2^j e 2^{j+1} é chamado de sinal de detalhamento no nível de resolução 2^j. Na seção anterior foi visto que as aproximações de $f(x)$ nos níveis de resolução 2^j e 2^{j+1} são, respectivamente iguais às suas projeções ortogonais nos subespaços vetoriais V_{2^j} e $V_{2^{j+1}}$. Assim, o sinal de detalhamento em 2^j é obtido através da projeção ortogonal de $f(x)$ em W_{2^j}.

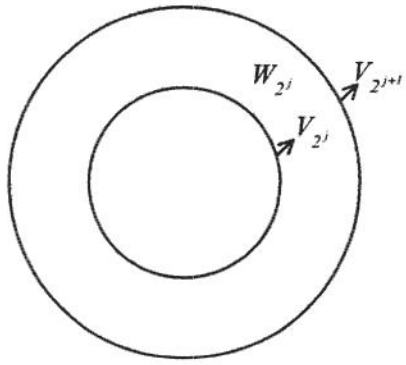


Fig. 6.3 – O espaço vetorial W_{2^j}

Usando um procedimento análogo ao da seção anterior, pode-se determinar a expressão do sinal de detalhamento $D_{2^j} \cdot f(x) \subset W_{2^j}$. Para isso é necessário a definição de uma base ortonormal para o subespaço W_{2^j}. Assim, seja assumindo a existência de uma função $\psi(x) \subset W_{2^j}$ de forma que uma família $\{\psi_{j,n}(x) = 2^{j/2} \cdot \psi(2^j x - n), n \in Z\}$ é uma base ortonormal para W_{2^j} [26][28], então será mostrada nessa seção a existência dessa função pela definição dela em função da função-escala e do filtro *h(n)*. Essa família será referenciada ao longo do texto simplesmente como $\{\psi_{j,n}\}$.

Dessa forma, como para qualquer $n \in Z$, a função $\psi_{j,n}(x) = 2^{j/2} \cdot \psi(2^j x - n)$ pertence a W_{2^j} que está contido em $V_{2^{j+1}}$, pode-se então expandir $\psi_{j,n}$ na base ortonormal de $V_{2^{j+1}}$:

$$\psi_{j,n}(x) = \sum_{k=-\infty}^{\infty} \langle \psi_{j,n}(y), \phi_{j+1,k}(y) \rangle \cdot \phi_{j+1,k}(x) \tag{6.28}$$

Fazendo a novamente a mudança de variáveis no produto interno, tem-se analogamente à seção anterior:

$$\langle \psi_{j,n}, \phi_{j+1,k} \rangle = 2^{j/2+(j+1)/2} \int_{-\infty}^{\infty} \psi(2^{-j} y) \cdot \phi(y - (k - 2n)) 2^{-j-1} dy$$

$$\langle \psi_{j,n}, \phi_{j+1,k} \rangle = 2^{-1/2} \langle \psi(2^{-j} u), \phi(u - (k - 2n)) \rangle \tag{6.29}$$

Assim, retornando à (6.28) tem-se:

$$\psi_{j,n}(x) = \sum_{k=-\infty}^{\infty} \langle \psi_{-1,0}(u), \phi_{0,k-2n}(u) \rangle \cdot \phi_{j+1,k}(x) \tag{6.30}$$

Aplicando o produto interno de *f(x)* com ambos os lados de (6.28), tem-se:

$$\langle f, \psi_{j,n} \rangle = \sum_{k=-\infty}^{\infty} \langle \psi_{-1,0}(u), \phi_{0,k-2n}(u) \rangle \cdot \langle f(u), \phi_{j+1,k}(u) \rangle \tag{6.31}$$

Novamente aplicando a teoria de filtros, definindo o filtro discreto *g(n)* de resposta

impulsiva em (6.32), pode-se encarar (6.30) e (6.31) como os processos de filtragem de (6.33) e (6.34).

$$g(m)=\langle \psi_{-j,0}(u),\ \phi_{0,m}(u)\rangle \quad \forall n \in Z \tag{6.32}$$

$$\psi_{j,n}(x)=\sum_{k=-\infty}^{\infty} g(k-2n)\cdot \phi_{j+1,k}(x) \tag{6.33}$$

$$\langle f,\psi_{j,n}\rangle=\sum_{k=-\infty}^{\infty} g(k-2n)\cdot\langle f,\phi_{j+1,k}\rangle \tag{6.34}$$

Para se obter a expressão que determina a função $\psi(x)$ basta fazer $j= n=0$ na equação (6.33), resultando em:

$$\psi(x)=\sum_{k=-\infty}^{\infty} g(k)\cdot 2^{1/2}\phi(2x-k) \tag{6.35}$$

Tomando-se a TF de ambos os lados de (6.33), tem-se:

$$\psi(w)=\sum_{k=-\infty}^{\infty} g(k)\frac{\phi(0,5w)}{2^{1/2}}e^{-i0,5wn}=\frac{\phi(0,5w)}{2^{1/2}}G(0,5w) \tag{6.36}$$

Deve-se notar que, diferentemente de $\phi(w)$, não há recursão para $\psi(w)$. $G(w)$ deve satisfazer as propriedades (6.37), (6.38) e (6.40) [31]:

$$|G(w)|^2+|G(w+\pi)|^2=2\ ,\forall w \tag{6.37}$$

Porat [31] também coloca que $G(w)$ e $H(w)$ satisfazem à seguinte relação:

$$H(w)G^*(w)+H(w+\pi)G^*(w+\pi)=0 \tag{6.38}$$

Para se saber se a função $\psi(w)$ é uma wavelet, é necessário fazer o teste da condição de admissibilidade. Para isso tem-se, calcula-se, primeiramente:

$$|H(0)|^2+|H(\pi)|^2=2+|H(\pi)|^2=2\Rightarrow|H(\pi)|^2=0 \tag{6.39}$$

e em seguida substitui-se (6.23) e (6.39) em (6.38) obtendo-se:

$$G(0)=0\Rightarrow\psi(0)=\frac{\phi(0)}{2^{1/2}}\cdot G(0)\ \Rightarrow\psi(0)=0 \tag{6.40}$$

Logo, $\psi(w)$ satisfaz a condição de admissibilidade e a família $\{\psi_{j,n}\}$ é uma base *wavelet*. Cncatenanado algumas equações anteriores pode-se obter a relação entre os filtros $g(n)$ e $h(n)$ que será a base da análise multi-resolução. Para isso, primeiro definindo:

$$H_a=H(w)\ ,\ G_a=G(w)\ ,\ H_b=H(w+\pi)\ ,\ G_b=H(w+\pi) \tag{6.41}$$

e em seguida substituindo (6.41) em (6.38), tem-se:

$$G_a/G_b = -\left(H_b/H_a\right)^* \tag{6.42}$$

Dividindo-se o primeiro lado de (6.42) por $\left[1+\left|G_a/G_b\right|^2\right]^{1/2}$ e o segundo lado por $\left[1+\left|H_b/H_a\right|^2\right]^{1/2}$, tem-se:

$$\frac{G_a/G_b}{\left[1+\left|G_a/G_b\right|^2\right]^{1/2}} = -\frac{\left(H_b/H_a\right)^*}{\left[1+\left|H_b/H_a\right|^2\right]^{1/2}} \tag{6.43}$$

Multiplicando matematicamente e substituindo (6.25) e (6.37) em (6.43), obtém-se:

$$G_a = -\frac{|H_a|G_b}{|G_b|H_a^*}\cdot H_b^* = -A(w)\cdot H_b^* \tag{6.44}$$

Analisando a definição de $A(w)$ pode-se observar que $|A(w)| = 1, \ \forall w$, ou seja, $A(w)$ é um filtro passa-tudo. Conseqüentemente $|G_a|=|H_b|$, o que se consiste na primeira relação entre os filtros e impõe que a magnitude do espectro de $g(n)$ é igual ao espectro de $h(n)$ deslocado de π. Como $h(n)$ é um filtro passa-baixas, $g(n)$ é, então, um filtro passa-altas.

Para analisar agora a questão da fase dos filtros, substitui-se (6.44) em (6.38) obtendo-se:

$$H(w)\left[-A(w)H^*(w+\pi)\right]^* + H(w+\pi)\left[-A(w+\pi)H^*(w+2\pi)\right]^* = 0 \tag{6.45}$$

$$A(w)+A(w+\pi)=0$$

Deve-se notar que a condição de admissibilidade está embutida em (6.44) (6.45). A escolha do filtro $A(w)$ é arbitrária, mas freqüentemente se usa $A(w)=e^{-iw}$, pois é uma forma simples, que obedece à $|A(w)|=1$ e à (6.45). Substituindo essa escolha em (6.44) tem-se:

$$G(w) = e^{-iw}\cdot H^*(w+\pi), \text{ portanto:}$$

$$g(n) = (-1)^{n-1}\cdot h^*(1-n) \tag{6.46}$$

A equação (6.46) expressa a relação entre os filtros $g(n)$ e $h(n)$ utilizados no processo de decomposição e síntese da TW [26]. Apesar de se poder utilizar qualquer par de filtros que obedeça às condições apresentadas nesta seção, mas é usual escolher $h(n)$ como sendo um filtro passa-baixas e, portanto, $g(n)$ como um filtro passa-altas. É interessante notar que determinando qualquer um dentre $h(n)$, $g(n)$, $\phi(x)$ e $\psi(x)$, o s demais são automaticamente determinados.

Analisando as equações (6,21), (6.36), (6.46) e o Teorema 6.1, pode-se notar que é possível especificar uma relação exata entre a wavelet $\psi(x)$ e a função-escala $\phi(x)$, provando, assim, que a pressuposição da existência da wavelet feita no início dessa seção está correta, permitindo que se enuncie o teorema [26]:

Teorema 6.2: "Seja $\{V_{2^j}\}$ o conjunto de subespaços vetoriais de uma análise de multi-resolução, $\phi(x)$ a função-escala e $h(n)$ o filtro correspondente. Seja $\psi(x)$ a função cuja TF é dada por:

$$\psi(w) = G(0{,}5w)\phi(0{,}5w)2^{-1/2} \tag{6.47}$$

onde $G(w) = e^{-iw} \cdot H^*(w+\pi)$ e, $\psi_{j,n}(x) = 2^{j/2} \cdot \psi(2^j x - n)$ então:

$$\psi_{j,n}(x) = 2^{j/2} \cdot \psi(2^j x - n)\ ,\ n \in Z \tag{6.48}$$

chamada no texto de $\{\psi_{j,n}\}$ é uma base ortonormal em W_{2^j} ".

O Teorema 6.2 estabelece uma relação entre a *wavelet* $\psi_{j,n}$ e a função-escala $\phi(x)$. Observe que a partir de um par de filtros que satisfaçam (6.23)-(6.26), pode-se encontrar a função-escala e a *wavelet*-mãe a partir de (6.21) e (6.47), respectivamente. De fato, é usualmente mais fácil começar com $h(n)$. Se $h(n)$ tem somente um número finito de entradas não-nulas, então $\phi(x)$, $\psi(x)$ e as *wavelets*-filhotes resultantes terão todas suporte compacto, i.e., serão não-nulas somente num pequeno intervalo em x [22][33].

Assim, o sinal de detalhamento de $f(x)$, $D_{2^j} \cdot f(x)$, com resolução 2^j é dado por:

$$D_{2^j} \cdot f(x) = \sum_{n=-\infty}^{\infty} \langle f(u), \psi_{j,n}(u) \rangle \cdot \psi_{j,n}(x) \tag{6.49}$$

e coeficientes desse sinal de detalhamento são obtidos em função dos seguintes produtos internos:

$$D^d_{2^j} \cdot f = \{ \langle f(u), \psi_{j,n}(u) \rangle\ ,\ n \in Z \} \tag{6.50}$$

onde $D^d_{2^j} \cdot f$ contém a diferença de informação entre $A^d_{2^{j+1}} \cdot f$ e $A^d_{2^j} \cdot f$

De (6.34) nota-se que é possível calcular o sinal de detalhamento $D^d_{2^j} \cdot f$ através da convolução de $A^d_{2^{j+1}} \cdot f$ com o filtro discreto $g(-n)$, e aplicando dizimação por um fator de dois no resultado, conforme esquematizado no ramo inferior da Fig. 6.2. O processo completo é definido por (6.11) e (6.49).

6.4 – Decomposição e Reconstrução usando *Wavelets* Ortogonais

A representação *wavelet* ortogonal $A_1^d \cdot f$ do sinal discreto f pode ser calculada através de sucessivas decomposições formando o conjunto de sinais composto pela aproximação $A_{2^{-J}}^d \cdot f$ com menor resolução e pelos detalhes $D_{2^j}^d \cdot f$ com resoluções 2^j, sendo $-J \leq j \leq -1$. Matematicamente, a representação wavelet de um sinal pode ser expressa por:

$$\left\{ A_{2^{-J}}^d \cdot f , \left\{ D_{2^j}^d \cdot f \right\}_{-J \leq j \leq -1} \right\} \tag{6.51}$$

Se o sinal original possui N amostras, então os sinais discretos $D_{2^j}^d \cdot f$ e $A_{2^j}^d \cdot f$ possuem $2^j . N$ amostras cada. Assim, a representação *wavelet* (6.51) apresenta o mesmo número de amostras que $A_1^d \cdot f$, graças à ortogonalidade.

Para calcular a representação wavelet pode-se utilizar diretamente o produto interno entre o sinal e as funções da base ortogonal, mas a forma mais eficiente é utilizar o algoritmo rápido de Mallat descrito na Seção 5.6, que se baseia na análise multi-resolução apresentada no presente capítulo.

Para realizar a reconstrução do sinal a partir da representação wavelet, deve-se notar que o espaço vetorial definido para a representação *wavelet* ortogonal é completo, pois o espaço W_{2^j} é o complemento ortogonal de V_{2^j} em $V_{2^{j+1}}$ e, portanto, $\left(\left\{ \phi_{j,n}(x) \right\}, \left\{ \psi_{j,n}(x) \right\} \right)_{n \in Z}$ é uma base ortonormal em $V_{2^{j+1}}$. Dessa forma, pode-se escrever $\phi_{j+1,n}(x)$ como:

$$\phi_{j+1,n}(x) = \sum_{k=-\infty}^{\infty} \left\langle \phi_{j+1,n} , \phi_{j,k} \right\rangle \cdot \phi_{j,k}(x) + \sum_{k=-\infty}^{\infty} \left\langle \phi_{j+1,n} , \psi_{j,k} \right\rangle \cdot \psi_{j,k}(x) \tag{6.52}$$

Calculando-se o produto interno de ambos os lados de (6.52) com $f(x)$, tem-se:

$$\left\langle f(x), \phi_{j+1,n}(x) \right\rangle = \sum_{k=-\infty}^{\infty} \left\langle \phi_{j+1,n} , \phi_{j,k} \right\rangle \cdot \left\langle f(x), \phi_{j,k}(x) \right\rangle + \sum_{k=-\infty}^{\infty} \left\langle \phi_{j+1,n} , \psi_{j,k} \right\rangle \cdot \left\langle f(x), \psi_{j,k}(x) \right\rangle \tag{6.53}$$

Em seguida, fazendo-se a mudança de variáveis em (6.53), obtém-se:

$$\left\langle \phi_{j+1,n}(u), \phi_{j,k}(u) \right\rangle = \left\langle \phi_{-1,0}(u), \phi_{0,n-2k}(u) \right\rangle = h(n-2k) \tag{6.54}$$

$$\left\langle \phi_{j+1,n}(u), \psi_{j,k}(u) \right\rangle = \left\langle \psi_{-1,0}(u), \phi_{0,n-2k}(u) \right\rangle = g(n-2k) \tag{6.55}$$

que, se substituídas em (6.53), permitem que ela seja reescrita em função dos filtros $g(n)$ e $h(n)$:

$$\underbrace{\langle f(x),\phi_{j+1,n}(x)\rangle}_{\text{componente de } A^d_{2^{j+1}}\cdot f} = \sum_{k=-\infty}^{\infty} h(n-2k)\cdot \underbrace{\langle f(x),\phi_{j,k}(x)\rangle}_{\text{componente de } A^d_{2^j}\cdot f} + \sum_{k=-\infty}^{\infty} g(n-2k)\cdot \underbrace{\langle f(x),\psi_{j,k}(x)\rangle}_{\text{componente de } D^d_{2^j}\cdot f} \qquad (6.56)$$

Em (6.56), nota-se que $A^d_{2^{j+1}}\cdot f$ pode ser construída inserindo-se um zero entre cada amostra de $A^d_{2^j}\cdot f$ e $D^d_{2^j}\cdot f$ (superamostragem) e convoluindo os resultados, respectivamente, com *h(n)* e *g(n)*. Deve-se notar que *cada* coeficiente de $A^d_{2^{j+1}}\cdot f$ é formado por *todos* os coeficientes de $A^d_{2^j}\cdot f$ e $D^d_{2^j}\cdot f$. O diagrama em blocos da reconstrução do sinal encontra-se na Fig. 6.4.

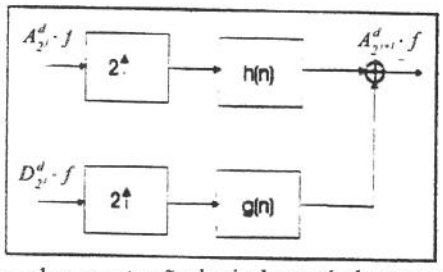


Fig. 6.4 - Esquema de reconstrução do sinal a partir da representação wavelet.

Comparando as Figs. 6.2 e 6.4, pode-se observar que no algoritmo de decomposição utilizam-se os filtros de resposta impulsiva *h(-n)* e *g(-n)*, enquanto na síntese utilizam-se *h(n)* e *g(n)*.

6.5 – Análise Multi-Resolução Bidimensional

Estendendo a análise multi-resolução para o caso 2D a partir de *wavelets* ortogonais [26][28], define-se o subespaço vetorial $V^2_{2^j}$ como o produto tensorial dado por:

$$V^2_{2^j} = V^1_{2^j} \otimes V^1_{2^j} \qquad (6.57)$$

Sendo $\{V^1_{2^j}\}$ a aproximação por multi-resolução em $L^2(R)$ dada na Seção 6.1, qualquer conjunto de subespaços vetoriais $\{V^2_{2^j}, j \in Z\}$ que satisfaçam as propriedades 6.1-6.6 para o caso bidimensional é chamado de *Aproximação por Multi-resolução em* $L^2(R^2)$. Ao longo do texto, esse conjunto será referenciado simplesmente por $\{V^2_{2^j}\}$. A aproximação de uma imagem *f(x,y)* no nível de resolução 2^j é a sua projeção no subespaço vetorial $V^2_{2^j}$.

Estendendo o Teorema 6.1 para o caso bidimensional, pode-se mostrar que há uma única função-escala $\phi(x,y) \in L^2(R^2)$ responsável pela formação de todas as bases ortogonais de todos os subespaços em questão [28]. Assim, a família $\{\phi_{j;n_x,n_y}(x,y)\}$ dada por:

$$\phi_{j;n_x,n_y}(x,y)=2^j\phi\left(2^j x-n_x,2^j y-n_y\right) \qquad ,(n_x,n_y)\in Z^2 \tag{6.58}$$

forma uma base ortonormal para cada $V_{2^j}^2$.

Sendo $\phi(x)$ a função-escala unidimensional referente a $\{V_{2^j}^1\}$, Mallat [34] mostra o caso em que a função-escala bidimensional $\phi(x,y)$ ortogonal é separável, podendo ser expressa por:

$$\phi(x,y)=\phi(x)\cdot\phi(y) \tag{6.59}$$

O mesmo ocorre para a base ortogonal em $V_{2^j}^2$:

$$\phi_{j;n_x,n_y}(x,y)=\phi_{j,n_x}(x)\cdot\phi_{j,n_y}(y) \qquad ,(n_x,n_y)\in Z^2 \tag{6.60}$$

Analogamente ao caso unidimensional, a aproximação de uma imagem $f(x,y)$ no subespaço vetorial $V_{2^j}^2$ é obtida por:

$$A_{2^j}\cdot f(x,y)=\sum_{n_x=-\infty}^{\infty}\sum_{n_y=-\infty}^{\infty}\left\langle f(u,v),\phi_{j;n_x,n_y}(u,v)\right\rangle\cdot\phi_{j;n_x,n_y}(x,y) \tag{6.61}$$

e a aproximação discreta de $f(x,y)$ na resolução 2^j é obtida em função dos seguintes produtos internos:

$$A_{2^j}^d\cdot f=\left\{\left\langle f(u,v),\phi_{j,n_x}(u)\cdot\phi_{j,n_y}(v)\right\rangle \qquad ,(n_x,n_y)\in Z^2\right\} \tag{6.62}$$

6.6 – Implementação de uma Transformada de Multiresolução 2D

As imagens utilizadas na prática são discretas, possuindo resolução máxima finita. Novamente por questão de normalização, será utilizada resolução máxima igual a 1. Assim, $A_1^d\cdot f(x,y)$ será a aproximação discreta de $f(x,y)$ nessa resolução.

Analogamente ao caso 1D, a família $\{\phi_{j;n_x,n_y}(x,y)\}$ é uma base ortonormal para cada $V_{2^j}^2$ e $\{\phi_{j;n_x,n_y}(x,y)\}\in V_{2^j}^2\subset V_{2^{j+1}}^2$. Logo, pode-se expandir $\{\phi_{j;n_x,n_y}(x,y)\}$ na base ortonormal de $V_{2^{j+1}}^2$:

$$\phi_{j,n_x,n_y}(x,y)=\sum_{k_x=-\infty}^{\infty}\sum_{k_y=-\infty}^{\infty}\left\langle\phi_{j;n_x,n_y}(u,v),\phi_{j+1;k_x,k_y}(u,v)\right\rangle\cdot\phi_{j+1;k_x,k_y}(x,y) \tag{6.63}$$

Fazendo a mudança de variáveis (análoga ao caso 1D) no produto interno de (6.7), tem-se que cada coeficiente resultante da projeção de $f(x,y)$ pode ser obtido por:

$$\left\langle f(x,y)\cdot\phi_{j;n_x,n_y}(x,y)\right\rangle=\sum_{k_y=-\infty}^{\infty}\sum_{k_x=-\infty}^{\infty}\left\langle\phi_{-1;0,0}(u,v),\phi_{0;(k_x-2n_x),(k_y-2n_y)}(u,v)\right\rangle\cdot\left\langle f(x,y)\cdot\phi_{j+1;k_x,k_y}(x,y)\right\rangle \tag{6.64}$$

Definindo o filtro discreto *h(l,m)* de resposta impulsiva em (6.65), pode-se encarar (6.64) como o processo de filtragem descrito em (6.66).

$$h(\ell,m)=\left\langle \phi_{-1;0,0}(u,v),\phi_{0;\ell,m}(u,v)\right\rangle =\left\langle \phi_{-1,0}(u),\phi_{0,\ell}(u)\right\rangle \cdot\left\langle \phi_{-1,0}(v),\phi_{0,m}(v)\right\rangle = h(\ell)\cdot h(m) \tag{6.65}$$

$$\left\langle f(x,y)\cdot\phi_{j;n_x,n_y}(x,y)\right\rangle = \sum_{k_y=-\infty}^{\infty} h(k_y-2n_y)\cdot\sum_{k_x=-\infty}^{\infty} h(k_x-2n_x)\cdot\left\langle f(x,y)\cdot\phi_{j+1;k_x,k_y}(x,y)\right\rangle \tag{6.66}$$

Deve-se notar que o filtro *h(l,m)* é separável nas direções *x* e *y*, podendo ser decomposto no produto de dois filtros digitais unidimensionais definidos em (6.16). A equação (6.66) é a expressão que, para (n_x,n_y) fixo, obtém cada coeficiente da projeção de *f(x,y)* na resolução 2^j a partir de todos os coeficientes da projeção de *f(x,y)* na resolução 2^{j+1}.

Esta equação mostra que $A_{2^j}^d \cdot f$ pode ser calculada através de filtragens sucessivas de $A_{2^{j+1}}^d \cdot f$ em ambas as direções. Primeiramente, $A_{2^{j+1}}^d \cdot f$ é filtrado na direção horizontal (direção das linhas, *x*) com o filtro *h(-n)* 1D, dizimando-se o resultado por um fator de 2 nessa direção (ou seja, descartando-se uma linha sim e outra não). A seguir, o resultado é filtrado na direção vertical (direção das colunas, *y*) com o mesmo *h(-n)*, seguido novamente pela dizimação por um fator de 2 na direção vertical (ou seja, descartando-se uma coluna sim e outra não). Esse processo encontra-se esquematizado no ramo superior da Fig. 6.5.

Todas as aproximações $A_{2^j}^d \cdot f$ para $j<0$ podem ser obtidas recursivamente tendo como ponto de partida $A_1^d \cdot f$ utilizando esse procedimento. Alternativamente, pode-se realizar primeiramente o processamento por colunas e depois por linhas, à semelhança do que ocorre com as demais transformadas bidimensionais. Essa opção também é válida para o sinal de detalhamento e para a transformação inversa, apresentadas na seqüência.

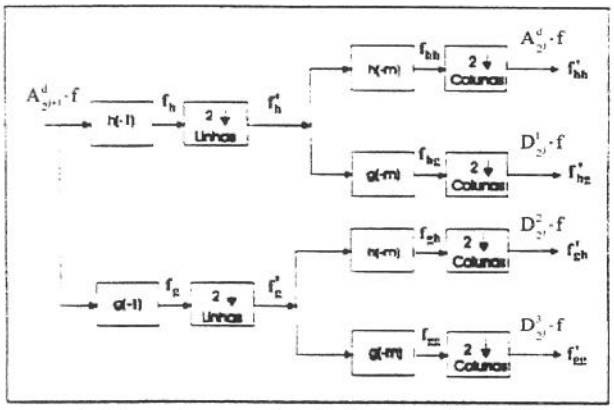


Fig. 6.5 - Diagrama de blocos de um estágio de decomposição wavelet de uma imagem.

6.7 - O sinal de Detalhamento 2D

Sendo $\{V_{2^j}^2\}$ a aproximação por multiresolução separável em $L^2(R^2)$, o detalhe no nível de resolução 2^j é a projeção de $f(x,y)$ no subespaço vetorial $W_{2^j}^2$ que, analogamente ao caso 1D, é o complemento ortogonal de $V_{2^j}^2$ em $V_{2^{j+1}}^2$.

Teorema 6.2: "Estendendo o Teorema 6.2 para o caso 2D, sendo a função-escala 2D separável $\phi(x,y)= \phi(x).\phi(y)$ e sendo $\psi(x)$ a *wavelet* unidimensional associada à $\phi(x)$, há 3 *wavelets* responsáveis pela formação das bases ortonormais [22][28]. As três *wavelets* bidimensionais são dadas por:

$$\begin{aligned}\psi^1(x,y) &= \phi(x)\cdot\psi(y)\\ \psi^2(x,y) &= \psi(x)\cdot\phi(y)\\ \psi^3(x,y) &= \psi(x)\cdot\psi(y)\end{aligned} \tag{6.67}$$

Assim, a família $\left\{ \psi^i_{j:n_x,n_y}(x,y),\ 1\le i\le 3\right\}$ dada em (6.68) é uma base ortonormal para cada $W_{2^j}^2$ ".

$$\left.\begin{aligned}\psi^1_{j:n_x,n_y}(x,y) &= 2^j\psi^1\left(2^j x-n_x\,,2^j y-n_y\right)\\ \psi^2_{j:n_x,n_y}(x,y) &= 2^j\psi^2\left(2^j x-n_x\,,2^j y-n_y\right)\\ \psi^3_{j:n_x,n_y}(x,y) &= 2^j\psi^3\left(2^j x-n_x\,,2^j y-n_y\right)\end{aligned}\right\},\ (n_x,n_y)\in Z^2 \tag{6.68}$$

Analogamente ao caso unidimensional e baseado no Teorema 6.2, a diferença de informação entre $A^d_{2^{j+1}}\cdot f$ e $A^d_{2^j}\cdot f$ é igual à projeção ortogonal de $f(x,y)$ no subespaço $W_{2^j}^2$ [26]:

$$\begin{aligned}D^1_{2^j}\cdot f &= \left\{ \left\langle f(x,y), 2^j\psi^1_{j:n_x,n_y}(x,y)\right\rangle \quad ,(n_x,n_y)\in Z\right\}\\ D^2_{2^j}\cdot f &= \left\{ \left\langle f(x,y), 2^j\psi^2_{j:n_x,n_y}(x,y)\right\rangle \quad ,(n_x,n_y)\in Z\right\}\\ D^3_{2^j}\cdot f &= \left\{ \left\langle f(x,y), 2^j\psi^3_{j:n_x,n_y}(x,y)\right\rangle \quad ,(n_x,n_y)\in Z\right\}\end{aligned} \tag{6.69}$$

que correspondem às três imagens de detalhes.

Assumindo a existência do conjunto de *wavelets* $\left\{ \psi^i_{j:n_x,n_y}(x,y),\ 1\le i\le 3\right\}$ separáveis, cada *wavelet* $\left\{\psi^i_{j:n_x,n_y}(x,y)\right\}$ pertence ao espaço $W_{2^j}^2\subset V_{2^{j+1}}^2$. Pode-se então expandir $\left\{\psi^i_{j:n_x,n_y}(x,y), 1\le i\le 3\right\}$ na base ortonormal de $V_{2^{j+1}}^2$:

$$\psi^i_{j:n_x,n_y}(x,y)=\sum_{k_x=-\infty}^{\infty}\sum_{k_y=-\infty}^{\infty}\left\langle \psi^i_{j:n_x,n_y}(u,v)\ ,\ \phi_{j+1:k_x,k_y}(u,v)\right\rangle\cdot\phi_{j+1:k_x,k_y}(x,y) \tag{6.70}$$

Aplicando uma mudança de variáveis, tem-se:

$$\psi^i_{j:n_x,n_y}(x,y)=\sum_{k_y=-\infty}^{\infty}\sum_{k_x=-\infty}^{\infty}\left\langle \psi^i_{-1:0,0}(u,v),\phi_{0:(k_x-2n_x),(k_y-2n_y)}(u,v)\right\rangle\cdot\phi_{j+1:k_x,k_y}(x,y) \tag{6.71}$$

Definindo o filtro discreto bidimensional de resposta impulsiva:

$$g^i(\ell,m)=\left\langle \psi^i_{-1:0,0}(u,v),\phi_{0:\ell,m}(u,v)\right\rangle \tag{6.72}$$

onde i indica o índice da *wavelet* utilizada, tem-se que para cada *wavelet* i definida em (6.11), existe um filtro separável correspondente que pode ser decomposto no produto de dois filtros 1D definidos em (6.16) e (6.32). Assim:

$$\begin{aligned}
g^1(\ell,m)&=\left\langle \phi_{-1,0}(u)\psi_{-1,0}(v),\phi_{0,\ell}(u)\phi_{0,m}(v)\right\rangle=\left\langle \phi_{-1,0}(u),\phi_{0,\ell}(u)\right\rangle\cdot\left\langle \psi_{-1,0}(v),\phi_{0,m}(v)\right\rangle=h(\ell)\cdot g(m)\\
g^2(\ell,m)&=\left\langle \psi_{-1,0}(u)\phi_{-1,0}(v),\phi_{0,\ell}(u)\phi_{0,m}(v)\right\rangle=\left\langle \psi_{-1,0}(u),\phi_{0,\ell}(u)\right\rangle\cdot\left\langle \phi_{-1,0}(v),\phi_{0,m}(v)\right\rangle=g(\ell)\cdot h(m)\\
g^3(\ell,m)&=\left\langle \psi_{-1,0}(u)\psi_{-1,0}(v),\phi_{0,\ell}(u)\phi_{0,m}(v)\right\rangle=\left\langle \psi_{-1,0}(u),\phi_{0,\ell}(u)\right\rangle\cdot\left\langle \psi_{-1,0}(v),\phi_{0,m}(v)\right\rangle=g(\ell)\cdot g(m)
\end{aligned} \tag{6.73}$$

Definidos os filtros pode-se obter o processo de filtragem em (6.18-6.20):

$$\left\langle f(x,y)\cdot\psi^1_{j:n_x,n_y}(x,y)\right\rangle=\sum_{k_y=-\infty}^{\infty}g(k_y-2n_y)\cdot\sum_{k_x=-\infty}^{\infty}h(k_x-2n_x)\cdot\left\langle f(x,y)\cdot\phi_{j+1:n_x,n_y}(x,y)\right\rangle \tag{6.74}$$

$$\left\langle f(x,y)\cdot\psi^2_{j:n_x,n_y}(x,y)\right\rangle=\sum_{k_y=-\infty}^{\infty}h(k_y-2n_y)\cdot\sum_{k_x=-\infty}^{\infty}g(k_x-2n_x)\cdot\left\langle f(x,y)\cdot\phi_{j+1:n_x,n_y}(x,y)\right\rangle \tag{6.75}$$

$$\left\langle f(x,y)\cdot\psi^3_{j:n_x,n_y}(x,y)\right\rangle=\sum_{k_y=-\infty}^{\infty}g(k_y-2n_y)\cdot\sum_{k_x=-\infty}^{\infty}g(k_x-2n_x)\cdot\left\langle f(x,y)\cdot\phi_{j+1:n_x,n_y}(x,y)\right\rangle \tag{6.76}$$

As equações (6.74)-(6.76) mostram que as diferentes imagens de detalhes $D^d_{2^j}\cdot f$ podem ser calculadas através de filtragens sucessivas de $A^d_{2^{j+1}}\cdot f$ em ambas as direções. Primeiramente $A^d_{2^{j+1}}\cdot f$ é filtrado na direção horizontal (linhas, x) com um filtro 1D, dizimando-se o resultado por um fator de 2 nesta direção. A seguir, o resultado é filtrado na direção vertical (colunas, y) com outro filtro 1D, seguido novamente pela dizimação por um fator de 2 na direção vertical. O filtro utilizado em cada direção depende do índice i da *wavelet* em questão.

Esse processo encontra-se esquematizado nos demais ramos da Fig. 6.5.

6.8 - Implementação de uma Representação *Wavelet* Ortogonal 2D

A representação *wavelet* ortogonal 2D de uma imagem original $A^d_1\cdot f$ pode ser calculada através de sucessivas decomposições de $A^d_{2^{j+1}}\cdot f$ em $A^d_{2^j}\cdot f$, $D^1_{2^j}\cdot f$, $D^2_{2^j}\cdot f$ e $D^3_{2^j}\cdot f$, sendo que

- $J \leq j \leq -1$. Um estágio completo deste algoritmo encontra-se na Fig. 6.5. A representação *wavelet* bidimensional pode, então, ser calculada através do cascateamento de dois algoritmos piramidais unidimensionais: o primeiro aplicado nas linhas da imagem, seguido pelo segundo aplicado às colunas.

Para qualquer $J > 0$, a representação *wavelet* 2D ortogonal de uma imagem $A_1^d \cdot f(x,y)$ é dada pelas $3J+1$ imagens discretas:

$$\left(A_{2^{-J}}^d \cdot f \ , \ \left\{D_{2^j}^1 \cdot f\right\}_{-J \leq j \leq -1} , \ \left\{D_{2^j}^2 \cdot f\right\}_{-J \leq j \leq -1} , \left\{D_{2^j}^3 \cdot f\right\}_{-J \leq j \leq -1} \right) \tag{6.77}$$

A imagem $A_{2^{-J}}^d \cdot f$ é a aproximação de $A_1^d \cdot f$ com nível de resolução 2^{-J}, correspondendo às freqüências mais baixas de $f(x,y)$. $\left\{D_{2^j}^l \cdot f\right\}_{l \leq i \leq 3}$ são as imagens de detalhamento para diferentes níveis de resoluções onde: $D_{2^j}^1 \cdot f$ corresponde às freqüências verticais altas (linhas horizontais na imagem), $D_{2^j}^2 \cdot f$ corresponde às freqüências horizontais altas (linhas verticais na imagem) e $D_{2^j}^3 \cdot f$ corresponde às freqüências altas em ambas as direções. Se a imagem original possui N *pixels*, cada imagem $A_{2^{-J}}^d \cdot f$, $D_{2^j}^1 \cdot f$, $D_{2^j}^2 \cdot f$ e $D_{2^j}^3 \cdot f$ da representação possui 2^{2j}. N *pixels*, totalizando os mesmos N *pixels* da imagem original.

A Fig. 6.6 ilustra a representação *wavelet* ortogonal 2D para três estágios ($J=3$).

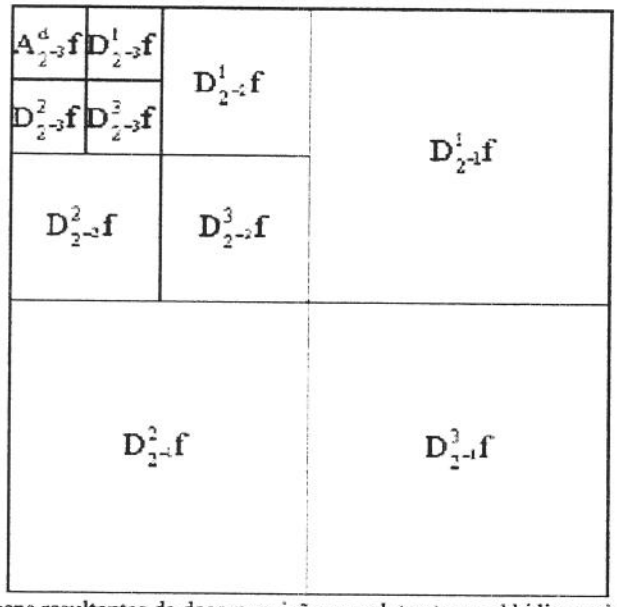


Fig. 6.6. Imagens resultantes da decomposição wavelet ortogonal bidimensional para J=3.

6.9 – Reconstrução a partir da Representação *Wavelet* Ortogonal 2D

Como $W_{2^j}^2$ é o complemento ortogonal de $V_{2^j}^2$ em $V_{2^{j+1}}^2$, o conjunto de funções $\left\{\left\{\phi_{j:n_x,n_y}(x,y)\right\}\left\{\psi_{j:n_x,n_y}^i(x,y),1\le i\le 3\right\}\right\}_{(n_x,n_y)\in Z}$ é a base ortonormal em $V_{2^{j+1}}^2$. Assim, pode-se escrever $\phi_{j+1:n_x,n_y}(x,y)$ como:

$$\begin{aligned}\phi_{j+1:n_x,n_y}(x,y)=&\sum_{k_y=-\infty}^{\infty}\sum_{k_x=-\infty}^{\infty}\left\langle\phi_{j+1:n_x,n_y}(u,v),\phi_{j:k_x,k_y}(u,v)\right\rangle\cdot\phi_{j:k_x,k_y}(x,y)+\\&+\sum_{i=1}^{3}\sum_{k_y=-\infty}^{\infty}\sum_{k_x=-\infty}^{\infty}\left\langle\phi_{j+1:n_x,n_y}(u,v),\psi_{j:k_x,k_y}^i(u,v)\right\rangle\cdot\psi_{j:k_x,k_y}^i(x,y)\end{aligned}\tag{6.78}$$

Calculando o produto interno de ambos os lados de (6.78) com *f(x,y)*, tem-se:

$$\begin{aligned}\left\langle f(x,y),\phi_{j+1:n_x,n_y}(x,y)\right\rangle=&\sum_{k_y=-\infty}^{\infty}\sum_{k_x=-\infty}^{\infty}\left\langle\phi_{j+1:n_x,n_y}(u,v),\phi_{j:k_x,k_y}(u,v)\right\rangle\cdot\left\langle f(x,y),\phi_{j:k_x,k_y}(x,y)\right\rangle+\\&+\sum_{i=1}^{3}\sum_{k_y=-\infty}^{\infty}\sum_{k_x=-\infty}^{\infty}\left\langle\phi_{j+1:n_x,n_y}(u,v),\psi_{j:k_x,k_y}^i(u,v)\right\rangle\cdot\left\langle f(x,y),\psi_{j:k_x,k_y}^i(x,y)\right\rangle\end{aligned}\tag{6.79}$$

Aplicando algumas substituições, tem-se:

$$\begin{aligned}\left\langle f(x,y)\cdot\phi_{j+1:n_x,n_y}(x,y)\right\rangle=&\sum_{k_y=-\infty}^{\infty}h(2n_y-k_y)\sum_{k_x=-\infty}^{\infty}h(2n_x-k_x)\cdot\left\langle f(x,y),\phi_{j:k_x,k_y}(x,y)\right\rangle+\\&+\sum_{k_y=-\infty}^{\infty}g(2n_y-k_y)\sum_{k_x=-\infty}^{\infty}h(2n_x-k_x)\cdot\left\langle f(x,y),\psi_{j:k_x,k_y}^1(x,y)\right\rangle+\\&+\sum_{k_y=-\infty}^{\infty}h(2n_y-k_y)\sum_{k_x=-\infty}^{\infty}g(2n_x-k_x)\cdot\left\langle f(x,y),\psi_{j:k_x,k_y}^2(x,y)\right\rangle+\\&+\sum_{k_y=-\infty}^{\infty}g(2n_y-k_y)\sum_{k_x=-\infty}^{\infty}g(2n_x-k_x)\cdot\left\langle f(x,y),\psi_{j:k_x,k_y}^3(x,y)\right\rangle\end{aligned}\tag{6.80}$$

Em (6.80) nota-se que para reconstruir $A_{2^{j+1}}\cdot f$ primeiro há a inserção de uma coluna de zeros entre cada coluna de $A_{2^j}^d\cdot f$, $D_{2^j}^1\cdot f$, $D_{2^j}^2\cdot f$, $D_{2^j}^3\cdot f$ seguida pela convolução das linhas resultantes com um filtro unidimensional. A seguir, há a inserção de uma linha de zeros entre cada linha, seguida pela convolução das colunas resultantes com outro filtro unidimensional. Os filtros utilizados são aqueles definidos em (6.16) e (6.32).

Deve-se notar que cada coeficiente de $A_{2^{j+1}}^d\cdot f$ é formado por todos os coeficientes de $A_{2^j}^d\cdot f$, $D_{2^j}^1\cdot f$, $D_{2^j}^2\cdot f$ e $D_{2^j}^3\cdot f$. O diagrama em blocos de um estágio da reconstrução da

imagem encontra-se na Fig. 6.7. A imagem $A_j^d \cdot f$ é reconstruída a partir da representação *wavelet* repetindo-se este processo para $-1 \leq j \leq -J$.

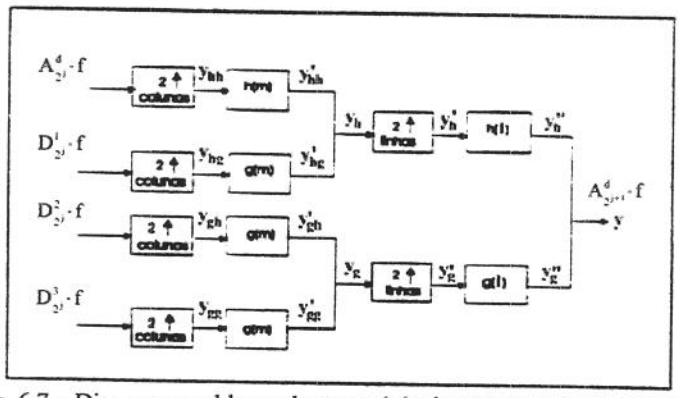


Fig. 6.7 – Diagrama em blocos de um estágio da reconstrução da imagem.

6.10 – Conclusões

Este capítulo apresentou a implementação da TWD (decomposição e síntese) para o caso ortogonal para os casos uni e bidimensionais. A importância fundamental desse capítulo reside na geração dos sinais de aproximação e de detalhes, que são os sinais efetivamente manipulados no processamento de sinais. Apresentada esta teoria ortogonal, pode-se relaxar as condições de operação, gerando a biortogonalidade, que sana deficiências do caso ortogonal e possibilita efetivamente a aplicação na compressão de imagens com excelente desempenho. A biortogonalidade consagrou o uso da TWD nos esquemas de compressão de imagens e é tema do capítulo seguinte.

Capítulo 7

Teoria de *Frames* e Biortogonalidade

A decomposição de sinais usando transformadas ortonormais apresenta propriedades muito importantes para o processamento digital de sinais como, por exemplo, a conservação de energia e a ausência de redundância entre os coeficientes transformados.

No entanto, no caso da aplicação da transformada *wavelet* em processamento digital de imagens, a utilização de transformadas ortogonais restringe a aplicabilidade dos processos de filtragem, pois limita as características desejáveis de suporte compacto, regularidade e simetria. *Wavelets* de suporte compacto são desejáveis, pois geram filtros de resposta impulsiva finita (FIR), que são estruturalmente mais simples e rápidos. Já a regularidade das *wavelets* garante a suavidade filtros assegurando uma melhor representação de uma maior quantidade de sinais. Contudo, suporte compacto é conflitante com a regularidade, conforme será abordado na Seção 7.2 [25] [33] [35].

Por fim, a simetria é uma característica fundamental porque gera filtros de fase linear e, em processamento digital de imagens, o uso de filtros de fase não-linear acarreta em distorções na imagem resultante.

Além desses fatores é sempre interessante que os filtros adotados permitam a reconstrução do sinal sem erros. No entanto, no caso das transformadas *wavelet*, Daubechies *et.al* [33] prova que não há filtro FIR ortonormal e simétrico trivial com propriedade de reconstrução exata. O único conjunto de filtros FIR simétricos com propriedade de reconstrução exata é o conjunto de filtros da *wavelet* de Haar [26], que por ser descontínua, possui má localização em freqüência.

A partir dessas restrições, a solução encontrada para gerar filtros simétricos, regulares, compactos e de reconstrução exata foi relaxar a condição de ortogonalidade, utilizando a teoria de *frames*, levando à biortogonalidade. A teoria de *frames* é abordada na seção 7.1.

7.1 - Teoria de *Frames*

Definição 7.1: "Seja $\{ g_k , k \in K\}$ um conjunto de funções em $L^2(R)$. Este conjunto é um *frame* se existirem números reais $0 < A \leq B < \infty$, tais que:

$$A\|f\|^2 \leq \sum_{k \in K} |\langle g_k , f \rangle|^2 \leq B\|f\|^2 \quad , \forall f \in L^2(R) \tag{7.1}$$

onde A e B são chamados de limitantes do *frame* [22]. O *frame* também pode ser definido como um

conjunto de funções, tais que qualquer função *f(x)* não-nula deve ter uma projeção não-nula em pelo menos uma delas".

Pela definição, pode-se afirmar que um *frame* é necessariamente um conjunto completo em $L^2(R)$, no entanto não há nenhuma restrição acerca da dependência linear entres as funções desse frame. Essas funções podem ser linearmente dependentes, gerando uma representação com redundância, ou linearmente independentes, representando o sinal sem redundância. Dessa forma, somente os *frames* cujas funções sejam linearmente independentes podem ser considerados uma base em $L^2(R)$, Nesse caso, o *frame* é também chamado de *frame* exato. Outra definição importante é a definição de *frame* estreito: um *frame* é dito ser estreito se seus limitantes A e B são iguais, indicando que a energia após a transformação é a energia de *f(x)* escalonada.

Considere, por exemplo R^2 e sua base ortonormal elementar $\vec{e_1} = [0,1]$ e $\vec{e_2} = [1,0]$. Definindo o conjunto de vetores $F = \{\vec{a_1}, \vec{a_2}, \vec{a_3}\}$ abaixo:

$$\vec{a_1} = \vec{e_1} \qquad \vec{a_2} = -\frac{1}{2}\vec{e_1} + \frac{\sqrt{3}}{2}\vec{e_2} \qquad \vec{a_3} = -\frac{1}{2}\vec{e_1} - \frac{\sqrt{3}}{2}\vec{e_2} \tag{7.2}$$

tem-se:

$$\sum_{j=1}^{3}\left|\left\langle f,\vec{a_j}\right\rangle\right|^2 = \left|\left\langle f,\vec{e_1}\right\rangle\right|^2 + \left|-\frac{1}{2}\left\langle f,\vec{e_1}\right\rangle + \frac{\sqrt{3}}{2}\left\langle f,\vec{e_2}\right\rangle\right|^2 + \left|-\frac{1}{2}\left\langle f,\vec{e_1}\right\rangle - \frac{\sqrt{3}}{2}\left\langle f,\vec{e_2}\right\rangle\right|^2$$

$$= \frac{3}{2}\left|\left\langle f,\vec{e_1}\right\rangle\right|^2 + \frac{3}{2}\left|\left\langle f,\vec{e_2}\right\rangle\right|^2 = \frac{3}{2}\left|\left\langle f,\vec{e}\right\rangle\right|^2 \tag{7.3}$$

O conjunto de vetores F obedece ao teorema 7.1 e, logo, é um *frame*. Como seus vetores são LD, F não é exato e não constitui uma base. No entanto $A=B=3/2$, de forma que o *frame* F é um *frame* estreito, e 3/2 é a sua razão de redundância [26].

Como o conceito de base engloba o conceito de base, é possível definir uma base ortonormal a partir dos conceitos de teoria de *frames*:

Lema 7.1: "Se o *frame* é estreito, se $A=B=1$ (conservação de energia aplicando a transformação), e se todas as suas funções $\{g_k, k \in K\}$ possuem norma unitária ($||g_k||=1$), o *frame* é uma base ortonormal [25]".

Para cada frame F, é possível determinado um operador T: $L^2(R) \rightarrow L^2(R)$, chamado operador frame, tal que se pode escrever a expansão em série:

$$T \cdot f(x) = \sum_{k \in K} \langle f, g_k \rangle \cdot g_k(x) \tag{7.4}$$

Em [31], Porat demonstrou que o operador T é sempre inversível e, portanto, a partir da

representação do sinal $Tf(x)$ pode-se recuperar o sinal original. Para encontrar a transformação inversa é necessário o seguinte Lema:

Lema 7.2: "Seja o conjunto de funções $\{ \gamma_k , k \in K\}$ obtido da seguinte forma [31]:

$$\gamma_k = T^{-1} g_k \tag{7.5}$$

Então pode-se definir o frame $\{ \gamma_k , k \in K\}$ como sendo o *frame* dual de $\{ g_k , k \in K\}$. Quando o *frame* é exato, os *frames* $\{ \gamma_k \}$ e $\{ g_k \}$ são biortogonais, logo:

$$\langle g_k , \gamma_l \rangle = \delta(k-l)". \tag{7.6}$$

De (7.4) e (7.5) segue o Teorema 7.1.

Teorema 7.1: "Para qualquer *frame* $\{ g_k \}$ e seu dual $\{ \gamma_k \}$, pode-se expressar qualquer função $f(x) \in L^2(R)$ através de qualquer uma das formas abaixo:

$$f(x) = \sum_{k \in K} \langle f, \gamma_k \rangle \cdot g_k(x) \tag{7.7}$$

$$f(x) = \sum_{k \in K} \langle f, g_k \rangle \cdot \gamma_k(x)". \tag{7.8}$$

O teorema 7.1 mostra que, dado um *frame* qualquer, é possível recuperar a função $f(x)$ a partir das suas projeções nas funções componentes de um *frame* e, portanto, que $f(x)$ pode ser representado como uma combinação linear das funções componentes do *frame*.

Deve-se notar que para realizar a transformada inversa descrita precisa-se então apenas calcular o *frame* dual [24]. Se o *frame* for estreito, $T=A.I$ (onde I é o operador identidade) e portanto $\gamma_k = A^{-1} g_k$, tornando trivial a obtenção do dual. Caso o *frame* não seja estreito, o dual é obtido de forma não trivial.

Na seção 7.3 será apresentada a forma de determinação dos *frames* duais para o caso de *frames* exatos e não estreitos (biortogonais).

7.2 - Ortogonalidade e Biortogonalidade

Aplicando a teoria de *frames* ao contexto da transformada wavelet, pode-se descrever o processo de decomposição e síntese de sinais por wavelets biortogonais. Para construir essa representação, basta que a *wavelet* $\psi(x)$ escolhida gere um *frame* $\{\psi_{j,n}\}$, isto é:

$$A\|f\|^2 \leq \sum_{j,k \in Z} |\langle f, \psi_{j,n} \rangle|^2 \leq B\|f\|^2 \ , \forall f \in L^2(R) \tag{7.9}$$

Dessa forma, pode-se afirmar que existe a transformada inversa para essa *wavelet* [25].

Assim, definindo-se $\{\tilde{\psi}_{j,n}\}$ como sendo o *frame* dual de $\{\psi_{j,n}\}$, pode-se reescrever as equações (7.4), (7.8) e (7.7) na seguinte forma:

$$T \cdot f(x) = \sum_{j,n \in Z} \langle f, \psi_{j,n} \rangle \cdot \psi_{j,n}(x) \tag{7.10}$$

$$f(x) = T^{-1} \cdot T \cdot f(x) = \sum_{j,n \in Z} \langle f, \psi_{j,n} \rangle \cdot T^{-1} \cdot \psi_{j,n}(x) = \sum_{j,n \in Z} \langle f, \psi_{j,n} \rangle \cdot \tilde{\psi}_{j,n}(x) \tag{7.11}$$

Logo, pela teoria de frames:

$$f(x) = \sum_{j,n \in Z} \langle f, \tilde{\psi}_{j,n} \rangle \cdot \psi_{j,n}(x) = \sum_{j,n \in Z} \langle f, \psi_{j,n} \rangle \cdot \tilde{\psi}_{j,n}(x) \tag{7.12}$$

A biortogonalidade é definida quando o *frame* $\{\psi_{j,n}\}$ é exato, e, portanto, obedecem à propriedade (7.13):

$$\langle \psi_{j,n}, \tilde{\psi}_{k,m} \rangle = \int_{-\infty}^{\infty} \psi_{j,n}(u) \cdot \tilde{\psi}_{k,m}(u)\, du = \begin{cases} 1, & se\ j = k\ e\ n = m \\ 0, & caso\ contrário \end{cases}, \quad j,n,k,m \in Z \tag{7.13}$$

Nesse caso, as *wavelets* são chamadas *wavelets* biortogonais. Se os *frames* forem também estreitos, então o conjunto formará uma base *wavelet* ortogonal e (7.11) fica:

$$f(x) = \sum_{j,n \in Z} \langle f, \psi_{j,n} \rangle \cdot A^{-1} \psi_{j,n}(x) = A^{-1} \sum_{j,n \in Z} \langle f, \psi_{j,n} \rangle \cdot \psi_{j,n}(x) \tag{7.14}$$

A partir de (7.14), para obter a ortonormalidade, basta fazer $A=1$, obtendo-se:

$$f(x) = \sum_{j,n \in Z} \langle f, \psi_{j,n} \rangle \cdot \psi_{j,n}(x) \tag{7.15}$$

É interessante notar que fazendo $\tilde{\psi}_{j,n}(x) = \psi_{j,n}(x)$ em (7.12) obtém-se (7.15) evidenciando o fato de as *wavelets* ortogonais serem um caso restrito das *wavelets* biortogonais. As principais diferenças entre ambas são [26]:

- A biortogonalidade permite a construção de *wavelets* e funções-escala simétricas (fase linear) e de reconstrução exata, o que se constitui numa de suas principais vantagens no processamento de imagens, conforme será mostrado no Capítulo 9.
- Os comprimentos dos filtros ortogonais *h(n)* e *g(n)* têm que ser pares e iguais entre si. Os filtros biortogonais não têm essa restrição, concedendo uma maior liberdade no projeto do filtro.
- Ao contrário das ortonormais, as wavelets biortogonais não conservam energia. Assim não é possível prever o valor exato da energia dos coeficientes transformados à partir da energia do sinal original. No entanto a energia varia dentro dos limites característicos de cada wavelet *A* e *B* mostrados na equação (7.9).

Continuando a descrição da transformada wavelet biortogonal, será apresentada na próxima seção a análise multi-resolução para o caso biortogonal.

7.3 - Análise de Multi-Resolução para *Wavelets* Biortogonais

Para o caso biortogonal, a análise de multi-resolução se assemelha à apresentada para o caso ortogonal, porém com uma implementação ligeiramente mais complicada. Com a biortogonalidade existe a definição de dois espaços vetoriais de aproximação $\{V_{2^j}\}$ e $\{\tilde{V}_{2^j}\}$, um par de funções-escala duais $\phi(x)$ e $\tilde{\phi}(x)$, dois espaços vetoriais de detalhamento $\{W_{2^j}\}$ e $\{\tilde{W}_{2^j}\}$ e finalmente um par de *wavelets* duais $\psi(x)$ e $\tilde{\psi}(x)$ [22].

Os subespaços V_{2^j} e $\tilde{V}_{2^j}$ obedecem individualmente às propriedades da seção 7.1, ou seja:

$$\begin{aligned} &\ldots \subset V_{2^{-1}} \subset V_{2^0} \subset V_{2^1} \subset V_{2^2} \subset \ldots \\ &\ldots \subset \tilde{V}_{2^{-1}} \subset \tilde{V}_{2^0} \subset \tilde{V}_{2^1} \subset \tilde{V}_{2^2} \subset \ldots \end{aligned} \tag{7.16}$$

Para qualquer $j \in Z$, $\{\phi_{j,n}\}$ e $\{\tilde{\phi}_{j,n}\}$ são bases não-ortogonais que geram, respectivamente, os subespaços V_{2^j} e $\tilde{V}_{2^j}$. Analogamente, essas bases são construídas a partir de deslocamentos, dilatações e compressões das funções-escala respectivas $\phi(x)$ e $\tilde{\phi}(x)$. A relação entre as funções-escala é dada por:

$$\langle \phi_{0,0}, \tilde{\phi}_{0,m} \rangle = \begin{cases} 1, & se\ m = 0 \\ 0, & caso\ contrário \end{cases} \tag{7.17}$$

O subespaço W_{2^j} é o complemento não-ortogonal de V_{2^j} em $V_{2^{j+1}}$. Já $\tilde{W}_{2^j}$ é o complemento não-ortogonal de $\tilde{V}_{2^j}$ em $\tilde{V}_{2^{j+1}}$, ou seja:

$$\begin{aligned} V_{2^{j+1}} &= V_{2^j} \oplus W_{2^j} \qquad V_{2^j}\ \textit{não é ortogonal à}\ W_{2^j} \\ \tilde{V}_{2^{j+1}} &= \tilde{V}_{2^j} \oplus \tilde{W}_{2^j} \qquad \tilde{V}_{2^j}\ \textit{não é ortogonal à}\ \tilde{W}_{2^j} \end{aligned} \tag{7.18}$$

Similarmente, os subespaços vetoriais W_{2^j} e $\tilde{W}_{2^j}$ obedecem à:

$$\begin{aligned} &\ldots \subset W_{2^{-1}} \subset W_{2^0} \subset W_{2^1} \subset W_{2^2} \subset \ldots \\ &\ldots \subset \tilde{W}_{2^{-1}} \subset \tilde{W}_{2^0} \subset \tilde{W}_{2^1} \subset \tilde{W}_{2^2} \subset \ldots \end{aligned} \tag{7.19}$$

e para qualquer $j \in Z$, $\{\psi_{j,n}\}$ e $\{\tilde{\psi}_{j,n}\}$ são bases não-ortogonais que geram, respectivamente, os subespaços W_{2^j} e $\tilde{W}_{2^j}$. Essas bases são construídas a partir de deslocamentos, dilatações e compressões das *wavelets*-básicas $\psi(x)$ e $\tilde{\psi}(x)$, respectivamente. As *wavelets*-básicas obedecem a:

$$\langle \psi_{0,0}, \tilde{\psi}_{0,m} \rangle = \delta(m) \tag{7.20}$$

Daubechies [9] também coloca a seguinte relação:

$$V_{2^j} \perp \tilde{W}_{2^j} \quad e \quad \tilde{V}_{2^j} \perp W_{2^j} \tag{7.21}$$

Assim, conhecendo apenas um dos subespaços e utilizando as equações (7.18) e (7.21) é possível determinar todos os restantes. Outra conclusão que pode ser demonstrada a partir de (7.21) é:

$$\langle \phi_{0,0}, \tilde{\psi}_{0,m} \rangle = \langle \tilde{\phi}_{0,0}, \psi_{0,m} \rangle = \delta(m) \tag{7.22}$$

Como conseqüência dessas propriedades pode-se escrever, usando a teoria de frames, a equação de decomposição da transformada wavelet biortogonal. Para isso, seja $\{\phi_{j,n}\}$ e $\{\tilde{\phi}_{j,n}\}$ os dois *frames* duais geradores dos subespaços V_{2^j} e $\tilde{V}_{2^j}$, portanto, a projeção da função $f(x) \in L(R^2)$ na base $\{\phi_{j,n}\}$ no nível de resolução 2^j para o caso biortogonal pode ser escrita como:

$$A_{2^j} \cdot f(x) = \sum_{n=-\infty}^{\infty} \langle f, \tilde{\phi}_{j,n} \rangle \cdot \phi_{j,n}(x) \tag{7.23}$$

e sua aproximação discreta :

$$A_{2^j}^d \cdot f = \{ \langle f, \tilde{\phi}_{j,n} \rangle, n \in Z \} \tag{7.24}$$

Analogamente, sendo $\{\psi_{j,n}\}$ e $\{\tilde{\psi}_{j,n}\}$ os dois *frames* duais geradores dos subespaços W_{2^j} e $\tilde{W}_{2^j}$, a projeção da função $f(x) \in L(R^2)$ na base $\{\psi_{j,n}\}$ no nível de resolução 2^j para o caso biortogonal é dada por:

$$D_{2^j} \cdot f(x) = \sum_{n=-\infty}^{\infty} \langle f, \tilde{\psi}_{j,n} \rangle \cdot \psi_{j,n}(x) \tag{7.25}$$

e o sinal de detalhamento discreto é:

$$D_{2^j}^d \cdot f = \{ \langle f, \tilde{\psi}_{j,n} \rangle \quad , n \in Z \} \tag{7.26}$$

Como $\{\tilde{\phi}_{j,n}\} \in \tilde{V}_{2^j} \subset \tilde{V}_{2^{j+1}}$, pode-se expandi-la na base de $\tilde{V}_{2^{j+1}}$, encarando-a como filtragem:

$$\tilde{\phi}_{j,n}(x)=\sum_{k=-\infty}^{\infty}\left\langle \tilde{\phi}_{j,n}\,,\phi_{j+1,k}\right\rangle\cdot\tilde{\phi}_{j+1,k}(x)=\sum_{k=-\infty}^{\infty}\left\langle \tilde{\phi}_{-1,0}(u),\phi_{0,k-2n}(u)\right\rangle\cdot\tilde{\phi}_{j+1,k}(x)=\sum_{k=-\infty}^{\infty}\tilde{h}(k-2n)\cdot\tilde{\phi}_{j+1,k}(x) \tag{7.27}$$

onde o filtro discreto de resposta impulsiva em questão é dado em (7.28):

$$\tilde{h}(m)=\left\langle \tilde{\phi}_{-1,0}(u),\phi_{0,m}(u)\right\rangle \tag{7.28}$$

Da mesma forma, como $\{\tilde{\psi}_{j,n}\}\in\tilde{W}_{2^j}\subset\tilde{V}_{2^{j+1}}$, pode-se expandi-la na base de $\tilde{V}_{2^{j+1}}$:

$$\tilde{\psi}_{j,n}(x)=\sum_{k=-\infty}^{\infty}\left\langle \tilde{\psi}_{j,n}\,,\phi_{j+1,k}\right\rangle\cdot\tilde{\phi}_{j+1,k}(x)=\sum_{k=-\infty}^{\infty}\left\langle \tilde{\psi}_{-1,0}(u),\phi_{0,k-2n}(u)\right\rangle\cdot\tilde{\phi}_{j+1,k}(x)=\sum_{k=-\infty}^{\infty}\tilde{g}(k-2n)\cdot\tilde{\phi}_{j+1,k}(x) \tag{7.29}$$

onde o filtro discreto de resposta impulsiva é dado em (7.30):

$$\tilde{g}(m)=\left\langle \tilde{\psi}_{-1,0}(u),\phi_{0,m}(u)\right\rangle \tag{7.30}$$

Convoluindo (7.27) e (7.29) com a função *f(x)*, tem-se respectivamente:

$$\left\langle f\,,\tilde{\phi}_{j,n}\right\rangle=\sum_{k=-\infty}^{\infty}\tilde{h}(k-2n)\cdot\left\langle f\,,\,\tilde{\phi}_{j+1,k}\right\rangle \tag{7.31}$$

$$\left\langle f\,,\tilde{\psi}_{j,n}\right\rangle=\sum_{k=-\infty}^{\infty}\tilde{g}(k-2n)\cdot\left\langle f\,,\,\tilde{\phi}_{j+1,k}\right\rangle \tag{7.32}$$

As equações (7.31) e (7.32) caracterizam o processo de decomposição da análise multi-resolução biortogonal. Analogamente ao caso ortogonal, pode-se calcular as aproximações discretas $A_{2^j}^d\cdot f$ e $D_{2^j}^d\cdot f$ através da convolução de $A_{2^{j+1}}^d\cdot f$ respectivamente com os filtros $\tilde{h}(-n)$ e $\tilde{g}(-n)$, seguidas pela dizimação por um fator de 2 (descartando uma amostra sim e outra não do resultado).

7.4 - Reconstrução a partir da Representação Wavelet Biortogonal

Como o subespaço W_{2^j} é o complemento não-ortogonal de V_{2^j} em $V_{2^{j+1}}$, através do conjunto de bases desses espaços pode-se gerar o espaço $V_{2^{j+1}}$. Assim, pode-se expandir tanto $\{\phi_{j+1,n}\}$ quanto $\{\tilde{\phi}_{j+1,n}\}$ em função do conjunto $\left(\{\phi_{j,n}\},\{\psi_{j,n}\}\right)_{n\in Z}$, mas como:

$$A_{2^{j+1}}^d\cdot f(x)=\sum_n\left\langle f(x),\tilde{\phi}_{j+1,n}\right\rangle\phi_{j+1,n} \tag{7.33}$$

é interessante começar o processo de reconstrução a partir da expansão de $\tilde{\phi}_{j+1,n}$. Assim:

$$\begin{aligned}\tilde{\phi}_{j+1,n}(x) &= \sum_{k=-\infty}^{\infty}\left\langle \tilde{\phi}_{j+1,n}(u),\phi_{j,k}(u)\right\rangle\cdot\tilde{\phi}_{j,k}(x)+\sum_{k=-\infty}^{\infty}\left\langle \tilde{\phi}_{j+1,n}(u),\psi_{j,k}(u)\right\rangle\cdot\tilde{\psi}_{j,k}(x)\\ &= \sum_{k=-\infty}^{\infty}\left\langle \tilde{\phi}_{0,n-2k}(u),\phi_{-1,0}(u)\right\rangle\cdot\tilde{\phi}_{j,k}(x)+\sum_{k=-\infty}^{\infty}\left\langle \tilde{\phi}_{0,n-2k}(u),\psi_{-1,0}(u)\right\rangle\cdot\tilde{\psi}_{j,k}(x)\end{aligned} \tag{7.34}$$

Definindo os filtros discretos de resposta impulsiva:

$$h(m)=\left\langle \phi_{-1,0}(u),\tilde{\phi}_{0,m}(u)\right\rangle \tag{7.35}$$

$$g(m)=\left\langle \psi_{-1,0}(u),\tilde{\phi}_{0,m}(u)\right\rangle \tag{7.36}$$

pode-se reescrever (7.34) como o processo de filtragem:

$$\tilde{\phi}_{j+1,n}(x)=\sum_{k=-\infty}^{\infty}h(n-2k)\cdot\tilde{\phi}_{j,k}(x)+\sum_{k=-\infty}^{\infty}g(n-2k)\cdot\tilde{\psi}_{j,k}(x) \tag{7.37}$$

Convoluindo (7.37) com a função $f(x)$, tem-se:

$$\underbrace{\left\langle f(x),\tilde{\phi}_{j+1,n}(x)\right\rangle}_{\text{componente de } A^{d}_{2j+1}\cdot f}=\sum_{k=-\infty}^{\infty}h(n-2k)\cdot\underbrace{\left\langle f(x),\tilde{\phi}_{j,k}(x)\right\rangle}_{\text{componente de } A^{d}_{2j}\cdot f}+\sum_{k=-\infty}^{\infty}g(n-2k)\cdot\underbrace{\left\langle f(x),\tilde{\psi}_{j,k}(x)\right\rangle}_{\text{componente de } D^{d}_{2j}\cdot f} \tag{7.38}$$

Em (7.38), pode-se notar que $A^{d}_{2j+1}\cdot f$ pode ser reconstruída através da inserção de uma amostra nula entre cada amostra de $A^{d}_{2j}\cdot f$ e $D^{d}_{2j}\cdot f$ (superamostragem) seguido da convolução do sinal resultante, respectivamente com $h(n)$ e $g(n)$. Assim, a reconstrução da análise de multi-resolução biortogonal é caracterizada por:

$$A^{d}_{2j+1}\cdot f(x)=A^{d}_{2j}\cdot f(x)+D^{d}_{2j}\cdot f(x) \tag{7.39}$$

O esquema da análise e síntese da TW biortogonal encontra-se na Fig. 7.1.

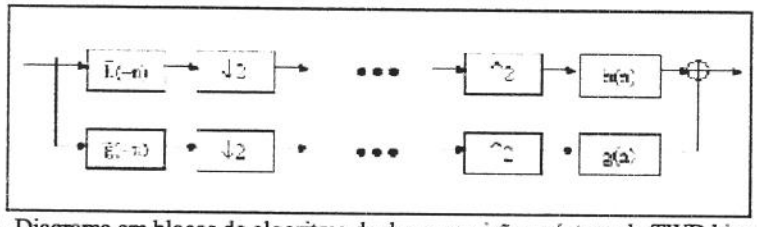


Fig. 7.1. Diagrama em blocos do algoritmo de decomposição e síntese da TWD biortogonal.

É interessante notar que, ao contrário do caso ortogonal, são utilizados filtros diferenciados para análise ($\tilde{h}(-n)$ e $\tilde{g}(-n)$) e síntese ($h(n)$ e $g(n)$). Dessa forma, os filtros biortogonais são mais flexíveis, fáceis de projetar e podem apresentar simetria, reconstrução exata e regularidade ao mesmo tempo, sendo, assim, muito apropriados para a aplicação em imagens [33].

7.5 - Conclusões

Neste capítulo foram apresentados os algoritmos de decomposição e síntese da TWD biortogonal, técnica que, estendida para o caso 2D, apresenta excelente desempenho nos esquemas de compressão de imagens. No próximo capítulo serão apresentadas as características das *wavelets* que são interessantes ao processamento de imagens e o algoritmo de codificação SPIHT (*Set Partitioned in Hierarchical Trees*).

Capítulo 8

Regularidade, Seleção das *Wavelets* e SPIHT

A importância da regularidade (suavidade) de uma *wavelet* está diretamente relacionada com a fidelidade na representação das funções nos subespaços vetoriais. Isso ocorre porque quanto mais irregular for a *wavelet*, apresentando, por exemplo, descontinuidades ou quebras, maior o espalhamento em freqüência da energia dos coeficientes transformados, o que acarreta erros e perda de qualidade visual em um sistema de compressão.

Neste Capítulo serão apresentadas as características desejáveis para as *wavelets* usadas na compressão de imagens, com destaque para a regularidade, culminando com a importância da biortogonalidade na compressão de imagens e nas características da TWD *spline* biortogonal 9-7, adotada por este trabalho.

Neste Capítulo também é apresentado o algoritmo SPIHT para codificação de imagens decompostas pela TWD que faz parte dos sistemas de codificação de imagens e vídeo desenvolvidos durante a tese de doutorado. O passo inicial é a apresentação das *wavelets* ortogonais de Daubechies.

8.1 - *Wavelets* de Daubechies

Daubechies apresentou em [35] um algoritmo para gerar um conjunto de *wavelets*-básicas, $\{\psi(x)|_N\}$ com características especiais. Caso $\psi(x)|_N$ fosse nula fora do intervalo *[0, 2N-1]*, seus primeiros *N* momentos também se anulavam, ou seja:

$$\int_{-\infty}^{\infty} x^n \psi(x)|_N \, dx = 0 \qquad n = 0,1,\ldots N \tag{8.1}$$

E seu número de derivadas contínuas é ≅ N/5. Isso descreve um grupo de funções bem comportadas, também chamadas de *wavelets regulares*.

Assim, o algoritmo de Daubechies apresentado a seguir permite a construção de filtros FIR *h(n)* que levam à *wavelets* $\psi(x)$ regulares, ortogonais e de suporte compacto. O algoritmo será apresentado em passos seqüenciais nos quais os filtros são representados em transformada Z de forma a simplificar a notação.

Passo 1) Escolher um número inteiro $N>0$, onde $2N$ é o comprimento do filtro $h(k)$;

Passo 2) Fazer

$$C(z)=\sum_{n=0}^{N-1}\binom{N-1+n}{n}\left(\frac{2-z-z^{-1}}{4}\right)^{n}+\left(\frac{2-z-z^{-1}}{4}\right)^{N}R\left(\frac{z+z^{-1}}{4}\right) \tag{8.2}$$

Onde $C(z)$ é um polinômio real e simétrico em z e z^{-1}. O polinômio $R(x)$ deve ser ímpar e escolhido de forma que $C(w)$ seja não negativo para qualquer w. Usar $R(x)=0$ satisfaz essa exigência, o que implicaria somente a necessidade de se escolher o N. Note-se que $C(w)$ é real, uma vez que $C(z)$ é simétrico;

Passo 3) Fatorar $C(z)$ como $C(z)=D(z).D(z^{-1})$, o que é possível de 2^k maneiras diferentes, sendo k o grau de $C(z)$;

Passo 4) Fazer $H(z)$ como:

$$H(z)=\left(\frac{1+z}{2}\right)^{N}\cdot D(z) \tag{8.3}$$

Passo 5) Calcular $G(w)$ a partir de $H(w)$ utilizando (6.46) e $\phi(x)$ utilizando (6.21) iterativamente com um numero de iterações (grau de precisão) arbitrário;

Passo 6) Finalmente, utilizar (6.47) para obter a wavelet-básica $\psi(x)$.

Por exemplo, se $R(x)=0$ e $C(z)$ tiver grau 2 haverá 4 $D(z)$'s diferentes. Dessa forma, haverá 4 filtros $h(k)$ possíveis e dessa forma, 4 funções-escala $\phi(x)$ diferentes, 4 $g(k)$ diferentes e daí 4 wavelets-básicas $\psi(x)$diferentes.

O parâmetro N é o índice de regularidade de $\phi(x)$ e $\psi(x)$. Quanto maior o valor de N, mais suave a *wavelet*-básica $\psi(x)$. Essas *wavelets* são conhecidas como "*wavelets* de Daubechies" e algumas delas encontram-se na Fig. 8.1. Curiosamente, $\psi(x)|_{N=1}$ é a *wavelet*-básica da transformada de Haar. A Tabela 8.1 mostra os coeficientes dos filtros $h(n)$ das *wavelets*-básicas com $N=3, 5, 7$ e 9.

Tabela 8.1 – coeficientes dos filtros $h(n)$ das *wavelets*-básicas com $N=3, 5, 7$ e 9.

N	Coefs									
3	h_0-h_5	,3327	,8069	,4599	-,1350	-,0854	,0352			
5	h_0-h_9	,1601	,6083	,7243	,1384	-,2423	-,0322	,0776	-,0062	-,0126	,0033
7	h_0-h_6	,0779	,3965	,7291	,4698	-,1439	,2240	,0713			
	h_7-h_{13}	,0806	-,0380	-,0166	0,126	,0004	-,0018	,0004			
9	h_0-h_8	.0381	,2438	,6048	,6573	,1332	-,2933	-,0968	,1485	,0307	
	h_9-h_{17}	-,0676	,0003	,0224	-,0047	-,0043	,0018	,0002	-,0003	,0000	

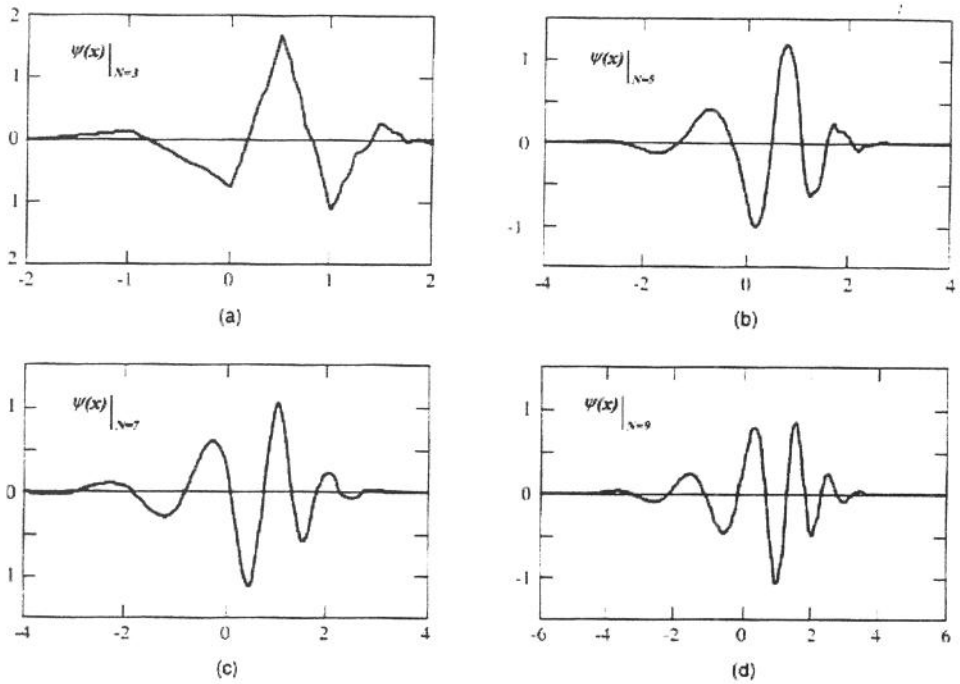


Fig. 8.1 – *Wavelets* de Daubechies para: (a) N=3; (b) N=5; (c)N=7 e (d)N=9.

8.2 - Seleção das *Wavelets*

A aplicação em processamento de imagens requer que a TW e, portanto seus filtros de implementação, apresentem certas características [22][25][26]:

Suporte compacto: a *wavelet* básica ideal deveria ser uma função oscilatória de curta duração (i.e., de suporte compacto), onde as translações diádicas de escalonamentos binários da função fossem ortonormais. A função de Haar obedece a essa premissa, porém, o mesmo não ocorre para muitas das *wavelets* disponíveis. Caso função-escala e *wavelet* possuam suporte compacto, as respostas impulsivas dos filtros correspondentes possuem comprimento finito (FIR), possibilitando algoritmos de rápida implementação.

Coeficientes racionais: Adotando filtros com coeficientes racionais, ou melhor, racionais diádicos, as operações de ponto flutuante são evitadas e um número menor de erros é introduzido na implementação.

Simetria: Filtros simétricos (também chamados de fase linear) tornam possível o cascateamento de bancos de filtros sem a necessidade de compensação de fase, evitando problemas nas bordas naturais das imagens.

Regularidade: Conforme apresentado, quanto mais regular a *wavelet* e a função-escala, maior o número de funções que podem ser representadas com fidelidade (sem o espalhamento da energia decorrente das descontinuidades das funções-base).

Contudo, nota-se que suporte compacto é conflitante com a regularidade, uma vez que quanto maior o comprimento do filtro, mais regular a *wavelet* para favorecer o processo de compressão; porém pior o aspecto de implementação do ponto de vista de velocidade. O objetivo passa a ser, portanto, encontrar o ponto ótimo com relação a esses aspectos conflitantes.

Além disso, teoricamente, os filtros adotados devem garantir a reconstrução do sinal sem erros. Infelizmente, Daubechies *et.al* [33] afirma que não há filtro FIR ortonormal e simétrico trivial com propriedade de reconstrução exata, independente da regularidade. Para contornar essas questões conflitantes, a condição de ortogonalidade é relaxada, gerando a biortogonalidade. Com ela, é possível preservar a simetria e a reconstrução exata e ainda obter alta regularidade. As *wavelets* biortogonais têm apresentado bom desempenho na compressão de imagens [33].

Deve ser notado que para transformadas sobrecompletas (como a TWC), as restrições a respeito das funções-base são suaves, enquanto para transformadas envolvendo pouca ou nenhuma redundância (como a TWD) as restrições são mais severas. Dentro das últimas, a TWD biortogonal apresenta maior flexibilidade na escolha da *wavelet*-básica a ser adotada. O cálculo das funções duais também não aumenta a complexidade computacional do processo.

Nesse contexto, o projeto de *wavelets* biortogonais tem sido uma área ativa de pesquisa. Vários autores têm catalogado filtros e suas *wavelets* biortogonais correspondentes.

8.3 - Implementação das *Wavelets* Biortogonais

Em [33][36] foram realizados extensos estudos matemáticos gerando técnicas para a construção de *wavelets* biortogonais. Dada a extensão e complexidade dessas técnicas e por fugirem ao escopo da tese, seus princípios serão somente sumarizados aqui.

Em [33] provou-se que uma alta regularidade pode ser obtida, tanto para ψ quanto para $\tilde{\psi}$, desde que se selecionem os filtros suficientemente longos. Em particular, se as funções ψ e $\tilde{\psi}$ forem, respectivamente, diferenciáveis $(p-1)$e $(\tilde{p}-1)$ vezes no tempo, então suas respostas em freqüências ($H(w)$ e $\tilde{H}(w)$) são divisíveis por $\left(1+e^{-jw}\right)^{p}$ e $\left(1+e^{-jw}\right)^{\tilde{p}}$, respectivamente. Logo, os filtros ($h(n)$ e $\tilde{h}(n)$) terão um comprimento maior que p e $\tilde{p}$, respectivamente [37].

Outro aspecto é que a divisibilidade de $\tilde{H}(w)$ por $\left(1+e^{-jw}\right)^{\tilde{p}}$ significa que ψ pode ter $\tilde{p}$

momentos nulos consecutivos [25], ou seja,

$$\int x^l \psi(x) dx = 0, \qquad para\ l = 0,1,....., \tilde{p} - 1 \tag{8.4}$$

Prova-se utilizando-se as séries de Taylor [25], que se ψ tem $\tilde{p}$ momentos nulos, os coeficientes $\langle f, \psi_{m,n} \rangle$ irão representar funções f que são $\tilde{p}$ vezes diferenciáveis com um alto potencial de compressão.

Muitos exemplos de *wavelets* biortogonais consideravelmente regulares podem ser construídos através dos algoritmos propostos por esses pesquisadores. Antonini [33] sugere que, dentro dos limites impostos pelo suporte compacto, procure-se escolher $\tilde{p}$ o maior possível para obter bom nível de regularidade.

Em termos de *H(w)* e $\tilde{H}(w)$, a expressão de reconstrução perfeita é dada por [33]:

$$H(w)\tilde{H}(w) + H(w+\pi)\tilde{H}(w+\pi) = 2 \tag{8.5}$$

Combinando-se as expressões (8.4) e (8.5) e levando-se em conta a imposição de divisibilidade de *H(w)* e $\tilde{H}(w)$ por $\left(1+e^{-jw}\right)^p$ e $\left(1+e^{-jw}\right)^{\tilde{p}}$, chega-se através de uma prova bastante extensa à [33]:

$$H(w)\tilde{H}(w) = \left(cos\left(w/2\right)\right)^{2l} \cdot \left[\sum_{q=0}^{l-1} \binom{l-1+q}{q} \left(sin\left(w/2\right)\right)^{2q} + \left(sin\left(w/2\right)\right)^{2l} R(w) \right] \tag{8.6}$$

Onde *R(w)* é um polinômio ímpar em *cos(w)*, e $2l = p + \tilde{p}$ (o fato de os filtros h e $\tilde{h}$ serem simétricos garante que $p + \tilde{p}$ seja par).

A partir de (8.6) é possível construir muitos exemplos de filtros biortogonais. De forma geral os passos são os seguintes: escolhe-se um *R(w)*; fatora-se *H(w)* e $\tilde{H}(w)$ gerando-se uma família; escolhe-se p e $\tilde{p}$ (comprimentos mínimos) gerando-se filtros diferentes. Nas próximas subseções serão mostrados três exemplos pertencentes a três diferentes famílias [37]. Na Tabela 8.2 são apresentados os coeficientes h_n e $\tilde{h}_n$ dos filtros passa-baixas destes três exemplos e nas Figs. 8.2, 8.3, 8.4 as respectivas *wavelets* e suas duais.

Uma vez obtidos h_n e $\tilde{h}_n$, para a implementação do processo completo basta utilizar as expressões, bastante semelhantes ao caso ortogonal, somente incluindo o conceito de dual [25]:

$$\tilde{g}(n) = (-1)^{n-1} \cdot h^*(1-n) \tag{8.5}$$

$$g(n) = (-1)^{n-1} \cdot \tilde{h}^*(1-n) \tag{8.6}$$

Tabela 8.2 – Coeficientes e comprimentos de filtros biortogonais *spline* e *spline* variantes

	n	0	±1	±2	±3	±4
Filtros *spline* $p=4$, $\tilde{p}=2$ e $l=3$ [9-3]	$2^{-l/2}.\ h_n$	45/64	19/64	-1/8	3/64	3/128
	$2^{-l/2}.\ \tilde{h}_n$	1/2	1/4	0	0	0
Filtros variante *spline* com comprimentos semelhantes $p=4$, $\tilde{p}=4$ e $l=4$ [9-7]	$2^{-l/2}.\ h_n$	0,602949	0,266864	-0,078223	-0,016864	0,026749
	$2^{-l/2}.\ \tilde{h}_n$	0,557543	0,295636	-0,028772	-0,045636	0
Filtros variante *spline* com comprimentos muito semelhantes $p=2$, $\tilde{p}=2$ e $l=2$ [5-7]	$2^{-l/2}.\ h_n$	0,6	0,25	-0,05	0	0
	$2^{-l/2}.\ \tilde{h}_n$	17/28	73/280	-3/56	-3/280	0

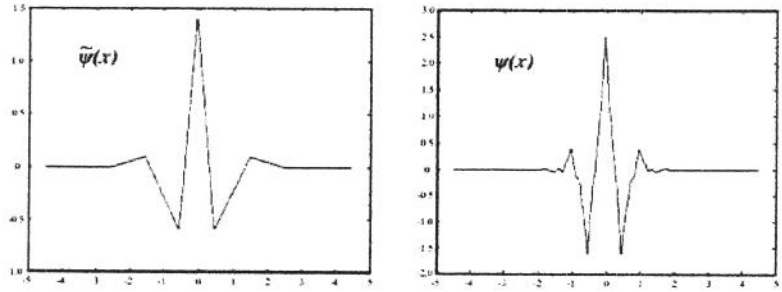


Fig. 8.2 – Wavelet biortogonal *spline* [9-3] e sua dual

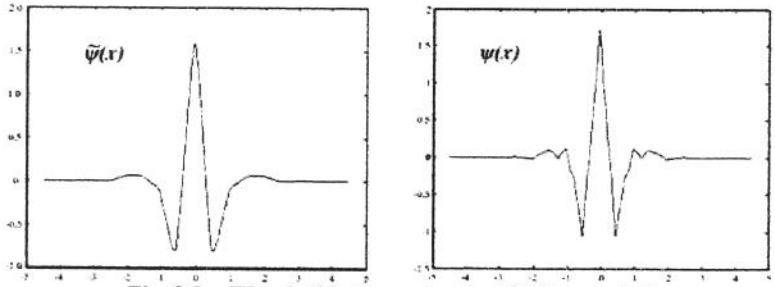


Fig. 8.3 – Wavelet biortogonal *spline* [9-7] e sua dual

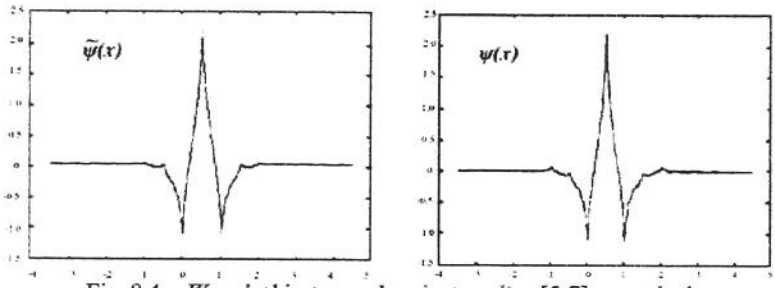


Fig. 8.4 – *Wavelet* biortogonal variante *spline* [5-7] e sua dual

8.3.1 - *Wavelets Spline* Biortogonais de *Cohen-Daubechies-Feauveau*

Fazendo $R(w) = 0$ em (8.4), e se a resposta do filtro passa-baixas dual for definida por:

$$\tilde{H}(w) = \left(\cos\left(w/2\right)\right)^{\tilde{p}} \cdot e^{-j\varepsilon w}, \text{ onde } \begin{cases} \varepsilon = 0, \text{ se } \tilde{p} \text{ é par} \\ \varepsilon = 1, \text{ se } \tilde{p} \text{ é ímpar} \end{cases} \tag{8.7}$$

Obtém-se a família de filtros conhecida como filtros *spline* [33], pois a função $\tilde{\phi}$ correspondente é *B-spline* ou *filtro binomial*, onde os $\tilde{h}$ são coeficientes binomiais. Os filtros *spline* são simétricos, suaves e tem coeficientes diádicos. A resposta em freqüência do filtro passa-baixas é dada por:

$$H(w) = \left(\cos\left(w/2\right)\right)^{2l-\tilde{p}} \cdot e^{\frac{j\varepsilon w}{2}} \left[\sum_{q=0}^{l-1} \binom{l-1+q}{q} \cdot \left(\sin\left(w/2\right)\right)^{2q} \right] \tag{8.8}$$

Na Tabela 8.2 foram apresentados os coeficientes h_n e $\tilde{h}_n$ dos filtros passa-baixas desse caso para $l=3$ e $\tilde{p}=2$. Observe que os filtros desse exemplo são simétricos com relação a 0, e possuem um comprimento ímpar. Essa é uma característica dos filtros *spline* [37].

8.3.2 - Variante de *Wavelets Spline* Biortogonais

Para se construir essa família de filtros faz-se $R(w) = 0$ em (8.6) e fatora-se o lado direito da expressão quebrando o polinômio de grau $l-1$ em $\sin(w/2)$, ou seja, em um produto de dois polinômios em $\sin(w/2)$. Na tentativa de tornar o comprimento do filtro h mais próximo do comprimento do filtro $\tilde{h}$ (facilitando a implementação), um dos dois polinômios resultantes (o que possui coeficientes reais) é definido como H e o outro como $\tilde{H}$ [37].

O exemplo na Tabela 8.2 é o filtro mais curto dessa família, com $l=p=4$. Trata-se da *spline* biortogonal [9-7], extremamente utilizada na literatura científica do estado da arte de compressão de imagens. Esse motivo, aliado às vantajosas características destas *wavelets* (suporte compacto, simetria, ótimo compromisso entre regularidade e rapidez dos algoritmos, reconstrução exata) motivaram a escolha da *spline* biortogonal [9-7] para os experimentos deste trabalho. A Fig. 8.5 ilustra a *wavelet* biortogonal spline [9,7] 2D.

8.3.3 - *Wavelets* próximas às Ortonormais.

Existem muitos exemplos de *wavelets* biortogonais para as quais $R(w) \neq 0$. Em particular, existe uma escolha de $R(w)$ para a qual os filtros em (8.6) estão muito próximos e conseqüentemente, muito próximos ao caso ortonormal. Um desses casos é o filtro piramidal laplaciano proposto em [32], para o qual $l=p=2$ e

$$R(w) = 48\cos\left(\frac{w}{2}\right)\Big/175 \tag{8.9}$$

Os coeficientes desse filtro podem ser consultados na Tabela 8.2. Os dois filtros biortogonais desse exemplo são ambos muito próximos de um filtro wavelet ortonormal de comprimento 6 apresentado em [38] denominados de *coiflets*. Sendo um filtro ortogonal, o *coiflet* não é simétrico para reconstrução exata. Os próximos filtros (l=p=4) de h e $\tilde{h}$ dessa família possuem comprimentos 9 e 15 e ambos são próximos ao *coiflet* de comprimento 12 [37].

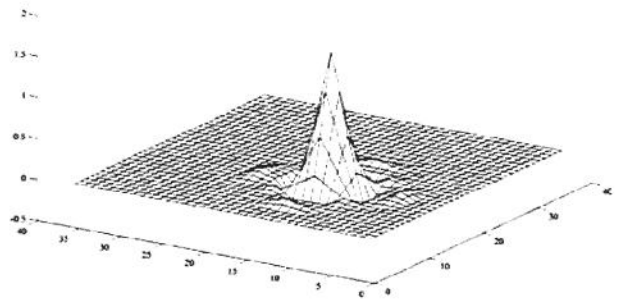


Fig. 8.5. Wavelet biortogonal *spline* [9,7] 2D

8.4 - SPIHT (*Set Partitioning in Hierarchical Trees*)

Dentre os algoritmos de codificação de coeficientes transformados por TWD encontrados na literatura científica, um dos que mais se destacam é o algoritmo SPIHT. Este algoritmo foi introduzido por Said e Pearlman [39][40][41] e se apresenta como uma versão bastante refinada do tradicional EZW (*Embedded Zerotree Wavelet Coding*) [42][43]. Alguns dos melhores resultados (excelente qualidade visual e valores de PSNR mais altos para mesmas razões de compressão) para uma ampla gama de imagens têm sido obtidos com esse algoritmo. Conseqüentemente, Rao e Yip [44] o apontam como sendo provavelmente o algoritmo de compressão de coeficientes-*wavelet* mais largamente utilizado, servindo de padrão básico de comparação para algoritmos subseqüentes.

Tanto o EZW quanto o SPIHT utilizam o conceito de árvores de particionamento. Como a decomposição concentra a energia nas sub-imagens de menor escala, em geral, os coeficientes possuem magnitudes maiores à medida que estão localizados mais próximos da raiz da árvore. Logo, se um dado coeficiente tiver magnitude menor que um dado limiar, sua descendência tende fortemente a apresentar magnitudes menores que esse mesmo limiar. Neste contexto, o EZW e o

SPIHT atuam como códigos de "embutimento" executados por passos de refinamento que possibilitam alta compressão e a transmissão progressiva dos dados.

No EZW, uma grande parcela de informação é transmitida a baixo custo quando uma sub-árvore inteira é declarada como não-significativa sendo representada com a simples "etiqueta" *zerotree*. O SPIHT usa um particionamento melhorado das árvores de forma a manter os coeficientes-*wavelet* insignificantes mais bem agrupados em *subconjuntos maiores* [45].

As decisões binárias de particionamento são transmitidas ao decodificador. A eficiência da codificação do mapa de significância pelo SPIHT é tal que a codificação aritmética das decisões binárias gera muito pouco ganho adicional na codificação [44].

Os limiares usados para testar a significância são potências de 2, de forma que em sua essência, o SPIHT termina por enviar a representação binária do valor inteiro dos coeficientes-*wavelet*. Da mesma forma que no EZW, a codificação do mapa de significância, ou o particionamento dos conjuntos e passo de ordenamento, é seguido por um passo de refinamento, no qual as representações dos coeficientes significativos são refinadas.

8.4.1 - Notação e Princípio de Funcionamento

Os coeficientes-*wavelet* são divididos em árvores originárias à partir da resolução mais baixa (banda I na Fig. 8.6). Estes coeficientes são agrupados em blocos de 2x2 *pixels* que, exceto para a banda I, são filhos de um coeficiente numa sub-banda de resolução mais baixa. Os coeficientes na sub-banda I são também agrupados em blocos de 2x2, entretanto, o coeficiente-*wavelet* do canto superior esquerdo de cada bloco dessa sub-banda não é tratado como um nó-raiz, não sendo vinculado a ele qualquer descendência. Essa estrutura pode ser observada na Fig 8.6 para uma DWT de 2 níveis [45].

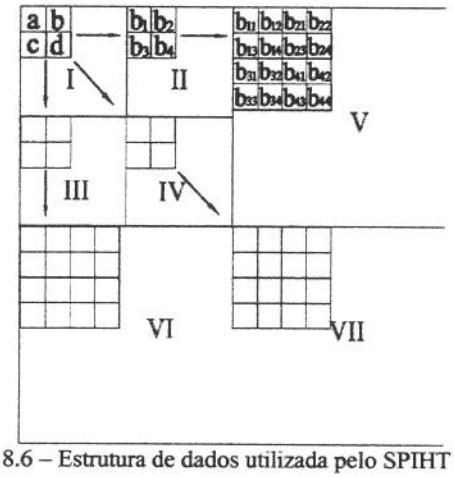


Fig. 8.6 – Estrutura de dados utilizada pelo SPIHT [46].

As árvores são particionadas em quatro tipos de conjuntos, que nada mais são do que conjuntos de coordenadas dos coeficientes:

1) $\mathcal{O}(i,j)$: conjunto de coordenadas dos "filhos" diretos de um dado coeficiente-*wavelet* da localização *(i,j)*. Como cada nó pode ter 4 filhos ou nenhum, $\mathcal{O}(i,j)$ possui tamanho zero ou 4, por exemplo, na Fig. 8.6 o conjunto $\mathcal{O}(1,2)$ consiste das coordenadas dos coeficientes b_1, b_2, b_3 *e* b_4 ;

2) $\mathcal{D}(i,j)$: conjunto de toda a descendência ("filhos, netos,...") de um dado coeficiente-*wavelet* da localização *(i,j)*. Por exemplo, na Fig 8.6 o conjunto $\mathcal{D}(1,2)$ consiste das coordenadas dos coeficientes b_1, ..., b_4 , b_{11}, ..., b_{14} ,..., b_{44} . Como o número de filhos pode ser zero ou 4, o tamanho de $\mathcal{D}(1,2)$ é zero ou a soma de potências de 4;

3) $\mathcal{H}$ *(i,j)* : conjunto de todos os nós-raiz (essencialmente banda I na Fig 8.6, salvo a exceção citada);

4) $\mathcal{L}$ *(i,j)* : conjunto de coordenadas de toda a descendência de um dado coeficiente-*wavelet* da localização *(i,j)*, com exceção dos seus "filhos" diretos. Na Fig 8.6 o conjunto $\mathcal{L}$ *(1,2)* consiste das coordenadas dos coeficientes b_{11}, ..., b_{14} ,..., b_{44} . Portanto:

$$\mathcal{L}(i,j) = \mathcal{D}(i,j) - \mathcal{O}(i,j) \tag{8.10}$$

Conforme já comentado, determinadas etapas do algoritmo SPIHT processam a informação referente a *conjuntos* de coeficientes-*wavelet*, e não somente aos coeficientes-*wavelet* individualmente. Para estas situações, um conjunto $\mathcal{D}(i,j)$ ou $\mathcal{L}(i,j)$ é dito significativo se *qualquer* de seus coeficientes integrantes apresentar magnitude maior do que o dado limiar. Esses limiares usados para testar a significância são potências de 2, de forma que em sua essência, o SPIHT termina por enviar a representação binária do valor inteiro dos coeficientes-*wavelet*.

Introduzidas as noções genéricas, segue-se a descrição do funcionamento do algoritmo em si. Primeiramente, esse algoritmo faz uso de três listas: LIP (Lista de *pixels* insignificantes), LIS (Lista de *conjuntos* insignificantes) e LSP (Lista de *pixels* significativos).

A LIP e a LSP conterão as coordenadas dos coeficientes, enquanto a LIS conterá as coordenadas das raízes dos conjuntos tipo $\mathcal{D}$ ou $\mathcal{L}$. Durante os passos de funcionamento, essas três listas são então analisadas para determinados valores de limiar, como será visto no exemplo.

8.4.2 - Acompanhamento

O algoritmo tem início com a determinação de valor do *threshold* inicial (T_0), dado por:

$$T_0 = 2^n \tag{8.11}$$

Onde, sendo c_{max} a magnitude máxima dos coeficientes-*wavelet* a codificar, n é dado por:

$$n = \lfloor \log_2 c_{\max} \rfloor \tag{8.12}$$

A LIP é inicializada contendo o conjunto $\mathcal{H}$. Os elementos de $\mathcal{H}$ que possuam descendência são também colocados em LIS, como sendo entradas do tipo $\mathcal{D}$. A LSP é inicializada como sendo vazia. Em cada passo, serão processados todos os componentes da LIP e a seguir os da LIS e esta composição determina o passo de codificação do mapa de significância. A seguir, o processamento da LSP determina o passo de refinamento.

O processamento da LIP consiste em examinar cada coordenada contida nela. Caso o coeficiente da coordenada seja insignificante (ou seja, menor do que 2^n), é transmitido um '*0*'. Caso o coeficiente da coordenada seja significativo (ou seja, maior ou igual a 2^n), é transmitido um '*1*' seguido por um bit representativo do sinal do coeficiente ('*0*' para positivo e '*1*' para negativo) e esse coeficiente é então movido para a LSP.

Após o processamento de todos os componentes da LIP, os conjuntos componentes da LIS são examinados. Caso o *conjunto* da coordenada seja insignificante (ou seja, todos os seus componentes menores do que 2^n), é transmitido um '*0*'. Caso o conjunto da coordenada seja significativo (ou seja, com pelo menos um elemento maior ou igual a 2^n), é transmitido um '*1*' e o que é feito em seqüência depende se trata-se de um conjunto do tipo $\mathcal{D}$ ou $\mathcal{L}$.

Caso trate-se de um conjunto tipo $\mathcal{D}$, é testada a significância de cada um dos seus 4 filhos, cujas coordenadas estão em $\mathcal{O}(i,j)$. Esses filhos são percorridos segundo uma varredura que privilegie o tipo de sub-imagem. Para cada filho significativo é transmitido um '*1*', a informação de sinal dele ('*0*' ou '*1*'), e então a coordenada dele é movida para a LSP. Para cada filho insignificante é transmitido um '*0*', e sua coordenada vai para a LIP.

Agora que as coordenadas de $\mathcal{O}(i,j)$ foram removidas do conjunto, o restante é o conjunto $\mathcal{L}(i,j)$, que se não for vazio, recebe a etiqueta $\mathcal{L}$ e é movido para o fim de LIS. Deve ser ressaltado que essa nova entrada da LIS terá que ser examinada durante o passo corrente. Caso $\mathcal{L}(i,j)$ seja vazio, a coordenada (i,j) é removida da lista.

Caso o conjunto seja do tipo $\mathcal{L}$, nós adicionamos cada coordenada de $\mathcal{O}(i,j)$ no fim de LIS como a raiz de um conjunto tipo $\mathcal{D}$. Deve ser ressaltado que essas novas entradas da LIS terão que ser examinadas também durante o passo corrente. A coordenada (i,j) é então removida da lista.

Uma vez processados todos os componentes de LIS, incluindo os novos termos, segue-se o passo de refinamento. Neste, cada coeficiente de LSP *do passo anterior* é examinado e a saída é o $(n+1)$ bit menos significativo de $|c_{i,j}|$. Os coeficientes de LSP adicionados no *passo corrente* não são

processados neste ponto porque, declarando-os como significativos, já informamos ao decodificador o valor do $(n+1)$ bit menos significativo de $|c_{ij}|$.

Após esses procedimentos, o primeiro passo é completado. O processo de codificação pode continuar, decrementando n de *1* e repetindo os passos explicados.

8.4.3 - Exemplo ilustrativo

Seja, por questão de simplicidade, a decomposição *wavelet* em um nível a seguir. Neste exemplo serão executados 3 passos de codificação e será gerado o *bitstream* que seria enviado ao decodificador. Em seguida, é exemplificada a decodificação deste *bitstream*.

54	-21	1	-8
-1	7	2	-3
1	-1	5	-7
4	0	11	4

Passo 1) $c_{max} = 54$. Logo, $n = 5$ e $T_0 = 32$. A inicialização das listas, portanto, é dada por:

LIP: $\{ (1,1) \rightarrow 54; (1,2) \rightarrow -21; (2,1) \rightarrow -1; (2,2) \rightarrow 7 \}$

LIS: $\{ (1,2)\mathcal{D}; (2,1)\mathcal{D}; (2,2)\mathcal{D} \}$

LSP: $\{ \}$

Bitstream inicial: -

Para facilitar a visualização, em parênteses são dadas as coordenadas e após a seta, o respectivo valor do coeficiente-*wavelet*. No *bitstream* também será, por vezes, colocado um certo espaçamento para facilitar a visualização. Inicialmente, examinando a LIP, sobre o coeficiente (1,1) transmite-se '*1*' (significativo, pois é maior do que 32) e '*0*' (significativo positivo) e suas coordenadas são movidas para a LSP. Os próximos 3 coeficientes de LIP são insignificantes com relação a 32, logo, são transmitidos um zero referente a cada um deles e eles permanecem em LIP.

Terminada a análise de LIP, segue-se o exame de LIS. Olhando para a descendência de (1,2) (etiquetada como $(1,2)\mathcal{D}$), que são os coeficientes 1, -8, -3 e 2 (segundo a ordem da varredura), observa-se que nenhum deles é significativo com relação a 32. Assim, para esse conjunto $(1,2)\mathcal{D}$, transmite-se um '*0*'. O mesmo acontece para os conjuntos $(2,1)\mathcal{D}$ e$(2,2)\mathcal{D}$ e, portanto, transmite-se um '*0*' para cada um deles.

A seguir, como não há coeficientes de LSP *do passo anterior*, não é feito o refinamento nesse passo. A situação das listas no fim deste primeiro passo é:

LIP: $\{ (1,2) \rightarrow -21; (2,1) \rightarrow -1; (2,2) \rightarrow 7 \}$

LIS: $\{ (1,2)\mathcal{D}; (2,1)\mathcal{D}; (2,2)\mathcal{D} \}$

LSP: $\{(1,1) \rightarrow 54\}$

Bitstream: 10 000 000 = 8 bits

Passo 2) Feito o decremento de 1 em n, $n = 4$ e $T = 16$.

Novamente, iniciando pelo exame da LIP. Há 3 coeficientes na LIP, sendo o primeiro significativo negativo, que gera a saída '11' e é movido para a LSP. Os dois seguintes são insignificantes e geram um '0' cada um.

Segue-se o exame de LIS. Novamente, nenhum dos coeficientes das descendências é significativo, então são gerados três '0's e segue-se o exame da LSP. A LSP contém o coeficiente $(1,1)$ que vale 54, cujo módulo é representado por '110110' em binário. Tomando o 5° bit menos significativo, gera-se um bit '1'.

A situação das listas no fim desse segundo passo é:

LIP: $\{ (2,1) \rightarrow -1; (2,2) \rightarrow 7 \}$

LIS: $\{ (1,2)\mathcal{D}; (2,1)\mathcal{D}; (2,2)\mathcal{D} \}$

LSP: $\{(1,1) \rightarrow 54; (1,3) \rightarrow -21\}$

Bitstream: 1100 000 1= 8 bits adicionais.

Passo 3) n = 3 e $T = 8$.

Seguindo a orientação dos procedimentos, pode-se verificar que a situação das listas e o *bitstream* transmitido no fim deste terceiro passo são:

LIP: $\{ (2,1) \rightarrow -1; (2,2) \rightarrow 7; (1,3) \rightarrow 1; (2,4) \rightarrow -3; (2,3) \rightarrow 2; (3,3) \rightarrow 5; (3,4) \rightarrow -7; (4,4) \rightarrow 4 \}$

LIS: $\{ (2,1)\mathcal{D} \}$

LSP: $\{(1,1) \rightarrow 54; (1,2) \rightarrow -21; (1,4) \rightarrow -8; (4,3) \rightarrow 11; \}$

Bitstream: 00 101100 0 100100 00 = 17 bits adicionais.

Claramente nota-se o aumento considerável no número de bits transmitidos à medida que o valor de T cai. Um breve exame no algoritmo EZW apresentado em [42] pode constatar de onde vem principalmente a economia em bits do SPIHT com relação ao EZW: o SPIHT reduz pela metade o número de bits que transporta o conceito de *zerotree* e os zeros isolados da última camada da DWT (de 2 para 1, já que no EZW existem 4 símbolos para codificar os coeficientes: *zerotree*,

signif.pos, *signif.neg*, e *zero isolado*) e estes coeficientes constituem a grande parcela de coeficientes-*wavelet*.

8.4.4 - Decodificação do exemplo

Segue-se a decodificação do *bitstream* gerado. No decodificador, o processamento também tem início com a mesma inicialização das listas:

LIP: { (1,1); (1,2); (2,1); (2,2) }

LIS: { (1,2)$\mathcal{D}$; (2,1)$\mathcal{D}$; (2,2)$\mathcal{D}$ }

LSP: { }

Tendo sido o valor de n inicial transmitido ao decodificador, sabe-se que $T_0=32$. Tendo recebido após o primeiro passo o *bitstream* 10 000 000, pode-se ver que o primeiro elemento da LIP é significativo positivo e os demais não, bem como suas descendências. Logo, a reconstrução inicial seria:

48	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

E as listas, seguindo o mesmo procedimento do codificador, seriam atualizadas para:

LIP: { (1,2); (2,1); (2,2) }

LIS: { (1,2)$\mathcal{D}$; (2,1)$\mathcal{D}$; (2,2)$\mathcal{D}$ }

LSP: {(1,1)}

Para o segundo passo, n é decrementado por 1 e o *bitstream* 1100 000 1 recebido é examinado. Há 3 entradas em LIP e os 2 primeiros bits (11) indicam que a primeira delas é significativa negativa e, portanto, será recuperada como $-1.5 \times 2^4 = -24$ e suas coordenadas serão movidas para a LSP. As duas restantes são insignificantes e permanecem na LIP. Na LIS, a indicação de três bits '0' consecutivos indica que nenhuma árvore é significativa e, portanto, também permanecem onde estão. Por fim será incluído o refinamento do elemento (1,1) presente na LSP. Como se recebeu um bit '1', então soma-se $2^{n-1} = 8$ ao coeficiente decodificado anteriormente.

Assim, o bloco decodificado até agora fica:

56	-24	0	0
0	0	0	0
0	0	0	0
0	0	0	0

E a situação das listas:

LIP: { (2,1); (2,2) }

LIS: { (1,2)$\mathcal{D}$; (2,1)$\mathcal{D}$; (2,2)$\mathcal{D}$ }

LSP: {(1,1); (1,2)}

Por fim, reduz-se $n = 3$ e toma-se o bitstream final: *00 101100 0 100100 00*. Pode-se ver que os elementos da LIP continuam insignificantes (primeiros dois bits '0'). Na LIS, no entanto, ocorrem mudanças. A árvore de descendentes de *(1,2)* é significativa (bit '1') e portanto será particionada. Dentre os descendentes diretos { *(1,3), (1,4), (2,4), (2,3)* }, o primeiro é insignificante ('0'), o segundo é significativo negativo ('11' recuperado como $-1{,}5x2^3=-12$), e os demais insignificante também ('00'). Os coeficientes são movidos para as listas de significância apropriadas e, como não há outros descendentes, somente retira-se a descendência de (1,2) da LIS.

Continuando a análise da LIS, tem-se que o segundo conjunto permanece insignificante e o terceiro conjunto é significante, possuindo um descendente significativo positivo na segunda posição da varredura. Esse coeficiente será decodificado com o valor +12. Varrendo a LSP temos dois bits '0' de refinamento, de forma que os módulos de ambos serão subtraídos dos coeficientes refinados o valor de $2^{n-1} = 4$. O estado final do bloco recuperado e das listas fica:

52	-20	0	-12
0	0	0	0
0	0	0	0
0	0	12	0

LIP: { (2,1); (2,2); (1,3); (2,4); (2,3); (3,3); (3,4); (4,4) }

LIS: { (2,1)$\mathcal{D}$ }

LSP: {(1,1); (1,2); (1,4); (4,3) }

Pode-se perceber a natureza da transmissão progressiva apresentada pelo SPIHT. Estudos mais recentes com relação a melhorias aplicadas sobre o SPIHT podem ainda ser encontrados em [46]-[49].

8.5 - Conclusões

Este capítulo abordou as características desejáveis para a TWD na compressão de imagens. Foram apresentadas questões como regularidade, suporte compacto e simetria; características altamente desejáveis e que podem ser simultaneamente satisfeitas através do uso de filtros biortogonais. Também foi apresentada a TWD *spline* biortogonal 9-7, adotada por este trabalho e suas características. Por fim, foi incluída uma descrição do algoritmo SPIHT, desenvolvido por Said

e Pearlman [39] que constou por um bom tempo como o melhor algoritmo de compressão dos avaliado no *site Bragzone* (http://links.uwaterloo.ca/bragzone.base.html) da Universidade de Waterloo.

Capítulo 9

Conceitos Básicos de Codificação de Vídeo

9.1 - Introdução:

Como citado anteriormente, um dos objetivos primários desse trabalho foi o desenvolvimento de um algoritmo de codificação de vídeo baseado na combinação de técnicas fractais e transformada wavelet. Dessa forma, tornou-se necessária a apresentação dos conceitos básicos de compressão de vídeo, com atenção especial aos algoritmos de compressão temporal, que se constituem na maior diferença entre as técnicas de codificação de imagens e vídeo.

Assim, nesse capítulo serão abordados conceitos básicos de vídeo, iniciando com o conceito de quadro (ou *frame*) e de seqüência de vídeo, passando por GOP (*Group of Pictures*), compressão temporal e estimação/compensação de movimento (ME/MC). A intenção é introduzir a terminologia da compressão de vídeo digital possibilitando uma melhor compreensão do algoritmo de codificação de vídeo desenvolvido nesse trabalho.

9.2 - Seqüência de Vídeo Digital: Conceitos

9.2.1 - Estrutura de Amostragem de Vídeo Digital

Um sinal de vídeo digital monocromático é composto por uma seqüência de imagens que representam uma amostragem temporal da projeção de uma cena (3D) sobre a superfície sensora (2D) da câmera de vídeo. Cada imagem da seqüência é chamada de Quadro ou *Frame*. Cada quadro é amostrado espacialmente obtendo-se uma estrutura matricial. Cada elemento dessa matriz é chamado de *Pixel* (*Picture Element*) e está associado a um valor de intensidade luminosa. Esse valor de intensidade luminosa é quantizado em 2^n valores inteiros, sendo normalmente $n=8$ *bits*. Esta representação é exemplificada na Fig. 9.1

Para seqüências de vídeo colorido, utilizam-se filtros cromáticos para separar os sinais luminosos em três cores primárias RGB (*Red-Green-Blue*) que são amostrados sob a mesma estrutura espaço-temporal, de forma que cada *pixel* passe a se representado em 24 bits, sendo 8 para cada campo de cor. Como nesse trabalho foram utilizadas apenas seqüências monocromáticas, um

estudo mais profundo sobre os processos de tratamento de vídeo colorido será apenas referenciado [50].

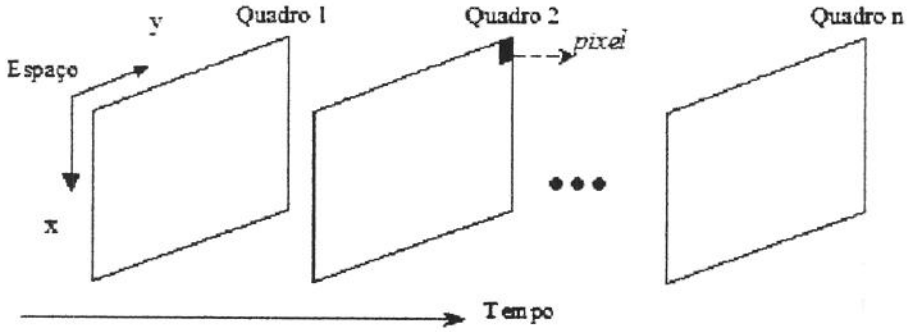


Fig. 9.1 - Exemplo de estrutura espaço-temporal de seqüência de vídeo

Além dessa estrutura, também costuma-se agrupar os pixels em blocos quadrados, normalmente de 8x8 pixels, e em macroblocos (MBs) que são o conjunto de 4 blocos formando uma matriz de 16x16 pixels. Tanto os blocos quanto os macroblocos são as unidades básicas de codificação definidas em cada uma das etapas necessárias à codificação de vídeo.

9.2.2 - Seqüência Progressiva ou Entrelaçada

Outro conceito muito comumente encontrado quando se trata de codificação de vídeo é o entrelaçamento. O entrelaçamento é a divisão de um quadro (*frame*) de uma seqüência em dois campos (*fields*), sendo que um campo contém apenas as linhas ímpares e o seguinte as linhas pares.

Essa separação de campos foi proposta originalmente para resolver um problema dos tubos de raios catódicos usados na recepção de televisão analógica. O problema que ocorria é que usando a transmissão de 30 quadros por segundo, os CRTs demoravam muito tempo para varrer a tela inteira progressivamente (linha a linha), de forma que a perda de intensidade do fósforo se tornava perceptível gerando um efeito de cintilação.

Para reduzir o efeito de cintilação, foi proposto que em vez de se atualizar a tela seqüencialmente (vídeo progressivo), ela fosse atualizada intercaladamente de forma que a perda de brilho de uma linha fosse parcialmente compensada pela da linha seguinte (vídeo entrelaçado).

Atualmente, além do grande desenvolvimento da tecnologia dos tubos CRT, o uso de vídeo digital possibilita que as linhas sejam armazenadas em memórias de forma que o efeito de cintilação já não é significativo, o entrelaçamento de vídeo deixou de ser obrigatório e a decisão da utilização ou não desse recurso passou a ser uma estratégia de codificação.

Assim, os padrões atuais de codificação de vídeo normalmente suportam tanto o formato

progressivo quanto o formato não-progressivo que inclui o vídeo entrelaçado e uma mistura de quadros entrelaçados com progressivos de forma a otimizar a qualidade do vídeo decodificado para a mesma taxa de compressão. Dessa forma, definiu-se para a codificação de vídeo digital que cada quadro (ou *frame*) é composto por um quadro não entrelaçado (*Frame-picture*) ou por dois campos de um quadro entrelaçado (dois *Field-pictures*).

Será apresentada em seguida a classificação dos tipos de quadros em função da compressão temporal.

9.2.3 - Compressão Temporal e Tipos de Quadros

Para explorar as redundâncias geradas a partir definição da estrutura de amostragem espaço-temporal foi desenvolvido o modelo de codificação de vídeo genérico que incluem as fases de compressão temporal (onde cada quadro é processado por macroblocos), compressão espacial (onde cada quadro é processado por blocos) e codificação entrópica. Normalmente, os algoritmos tradicionais de codificação, como o MPEG-2, utilizam a estrutura indicada na Fig. 9.2 abaixo:

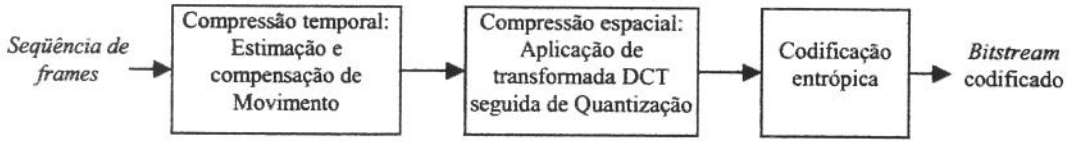


Fig. 9.2 - Esquema básico de um codificador híbrido de vídeo digital tradicional.

De acordo com o tipo de compressão temporal (estimação/compensação de movimento) a ser utilizada, classifica-se cada um dos quadros da seqüência em Quadro I *ou* Quadro P. Os Quadros I são os quadros codificados sem compensação de movimento [51] e, portanto, nenhum dos seus MBs sofre compressão temporal sendo assim denominados macroblocos *Intra* ou MB-*Intra*. Esses quadros serão usados como referência de estimação para os quadros subseqüentes.

Nos Quadros P, os MBs podem ser codificados usando estimação/compensação de movimento [51], ou seja, são codificados diferencialmente com relação a um quadro referência anterior já codificado. Dessa forma, procura-se eliminar a redundância entre os quadros codificados de forma a melhorar o desempenho da compressão.

No caso de codificadores de vídeo mais avançados, como o MPEG-2, também se define mais um tipo de quadro, o chamado Quadro B. Nesse tipo de quadro, cada MB é procurado por similares tanto nos quadros anteriores como nos subseqüentes, de forma que se extraia ainda mais redundância temporal. No entanto, a utilização de Quadros B obriga ao codificador executar uma mudança na ordem dos quadros, criando uma separação entre a ordem dos quadros na entrada e na

saída do codificador. Esta separação obriga a realização de um reordenamento dos quadros, aumentando a complexidade do codificador e do decodificador.

Um dos problemas da utilização desse tipo de codificação preditiva é que, se ocorrer um erro em um quadro que é usado como referência para a predição, esse erro tende a se propagar causando o efeito de *drift*. Para reduzir o efeito de *drift*, é necessário que de tempos em tempos o sistema de referências seja atualizado sem predição de forma que se anule a propagação. Assim, dentro da seqüência de quadros de vídeo pode-se definir grupos de quadros que utilizam a mesma referência inicial. Esses grupos são chamados de GOPs – *Group of Pictures* – e são caracterizados pela presença de um Quadro I como primeiro quadro do grupo.

No próximo item serão introduzidos alguns conceitos básicos sobre teoria de codificação e compressão de sinais digitais voltados para a codificação de vídeo.

9.2.4 - Compressão Com Perdas ou Sem Perdas

A informação visual pode ser digitalizada, ou seja, representada por uma quantidade finita de dados digitais, ou no caso de imagens variantes no tempo (vídeo), por uma razão finita de dados. Fato é que a representação digital normalmente contém uma quantidade considerável de informação redundante ou irrelevante, que pode ser removida por diversos métodos, sem que haja considerável prejuízo da qualidade das imagens.

A informação redundante está relacionada à correlação e interdependência dos dados, podendo ser removida sem deteriorar a qualidade da imagem, uma vez que pode ser completamente predita a partir da informação restante da imagem.

A informação irrelevante está relacionada às características que podem ser removidas sem que o sistema-destino, no caso o HVS (*Human Visual System*), perceba. Esse tipo de informação não pode ser recuperado após sua eliminação, portanto deve ser removida com critério para evitar que ocorram perdas significativas na qualidade da imagem.

As técnicas de compressão são então classificadas de acordo com o tipo de informação que removem: Compressão sem Perdas (*Lossless Compression*), que remove a informação redundante, e Compressão com Perdas (*Lossy Compression*), que remove a informação irrelevante.

Pode-se dizer que a compressão é um processo que gera uma representação digital compacta do sinal através da exploração da informação redundante e/ou irrelevante, preservando os níveis de qualidade necessários a uma dada aplicação.

Numa seqüência de vídeo, a redundância e a irrelevância aparecem tanto entre *pixels* de quadros próximos temporalmente quanto entre *pixels* no mesmo quadro. Assim, o MPEG-2 divide a

compressão em duas etapas:

- Compressão Temporal (Sem perdas): etapa que explora a redundância entre *pixels* de quadros consecutivos/ próximos.
- Compressão Espacial (Com perdas): etapa que explora a redundância e a irrelevância entre *pixels* vizinhos no mesmo quadro.

9.3 - O Padrão MPEG-2: Introdução

Em 1988, em resposta a uma necessidade crescente por um formato comum para a codificação e armazenamento de vídeo digital, a ISO (*The International Organization for Standardization*) convocou um grupo de especialistas conhecido como MPEG (*Moving Pictures Expert Group*). Em 1990 o MPEG gerou a segunda fase de seu trabalho: o padrão ISO-13818 Codificação Genérica de Imagens de Vídeo e Informação de Áudio Associado, também conhecido como padrão MPEG-2, capaz de gerar taxas de compressão da ordem de 50 a 75:1 [43] [51][52].

Os sistemas HDTV americano, europeu e japonês adotaram o padrão MPEG-2 como subsistema de compressão e codificação, somente apresentando algumas diferenças entre si quanto aos parâmetros utilizados. Talvez a diferença mais significativa entre eles seja a possibilidade de codificação hierárquica proporcionada somente pelos sistemas europeu e japonês.

O MPEG-2 é um padrão genérico, ou seja, independente da aplicação. Ao invés de especificar exatamente as operações do codificador, permite uma ampla gama de opções para a codificação. Este sistema especifica sim, a sintaxe do *bitstream* codificado e o processo de decodificação. Entende-se por *bitstream*, uma série ordenada de bits que forma a representação codificada dos dados.

Este padrão apresenta uma ampla gama aplicações basicamente relacionadas aos meios de comunicação digital, por exemplo: TV e HDTV (transmissão *broadcasting*, via satélite ou via cabo), vídeo em meios de armazenamento digital (CD-ROM, CD de alta densidade, DVD), vídeo em computadores (e-mail vídeo e sistemas de informação multimídia), vídeo transmitido através de redes (ATM, Ethernet e LAN's), editoração eletrônica, entre outros.

O MPEG-2 compreende 11 partes, das quais as principais são: *Vídeo, Systems, Áudio* e *Compliance Testing;* sendo *Vídeo* a parte relevante para esta pesquisa de doutorado. Como a utilização de todas as funcionalidades do MPEG-2 pode inviabilizar a implementação prática na maioria das aplicações devido à complexidade, o padrão foi dividido em perfis (*profiles*) e níveis (*levels*).

Um perfil é um subconjunto definido da sintaxe completa do *bitstream*. Os níveis estabelecem limitantes superiores para os parâmetros utilizados. Para cada par perfil/nível é definida uma sintaxe para o *bitstream* codificado e as necessidades de codificação. Os perfis (P) são: *Simple*, *Main*, *4:2:2*, *SNR*, *Spatial*, *High* e *Multiview*, como pode ser visto na Fig. 9.3. Para cada perfil, os 4 níveis (L) são: *Low*, *Main*, *High-1440* e *High* [51].

Levels		Profiles (Nonscalable)		
		Simple 4:2:0 (I,P)	Main 4:2:0 (I,B,P)	4:2:2 (I,/I,B,P)
High	Max resolution/ rate (Hz)	N/A	1920 x 1152/60	N/A
	Bitrate (Mbits/s)	N/A	80	N/A
High-1440	Max resolution/ rate (Hz)	N/A	1440 x 1152/60	N/A
	Bitrate (Mbits/s)	N/A	60	N/A
Main	Max resolution/ rate (Hz)	720 x 576/30	720 x 576/30	720 x 608/30
	Bitrate (Mbits/s)	15	15	50
Low	Max resolution/ rate (Hz)	N/A	352 x 288/30	N/A
	Bitrate (Mbits/s)	N/A	4	N/A

Bit rate: a razão na qual o bitstream codificado é entregue do meio de armazenamento à entrada do decodificador [16]

Frame rate: a razão na qual os *frames* saem do processo de decodificação [16].

As áreas com "N/A" indicam que não há restrições.

Levels		Profiles (Scalable)			
		SNR 4:2:0	Spatial 4:2:0	High 4:2:0, 4:2:2	Multiview 4:2:0
High	Enhancement (Auxiliary)	N/A	N/A	1920 x 1152/60	1920 x 1152/60
	Lower (Base)	N/A	N/A	960 x 576/30	1920 x 1152/60
	Bitrate (Mbits/s)	N/A	N/A	100 (all layers) 80 (base + mid) 25 (base layer)	130 (all layers) 50 (auxiliary) 80 (base layer)
High-1440	Enhancement (Auxiliary)	N/A	1440 x 1152/60	1440 x 1152/60	1920 x 1152/60
	Lower (Base)	N/A	720 x 576/30	720 x 576/30	1920 x 1152/60
	Bitrate (Mbits/s)	N/A	60 (all layers) 40 (base + mid) 15 (base layer)	80 (all layers) 60 (base + mid) 20 (base layer)	100 (all layers) 40 (auxiliary) 60 (base layer)
Main	Enhancement (Auxiliary)	720 x 576/30	N/A	720 x 576/30	720 x 576/30
	Lower (Base)	-	N/A	352 x 288/30	720 x 576/30
	Bitrate (Mbits/s)	15 (all layers) 10 (base layer)	N/A	20 (all layers) 15 (base + mid) 4 (base layer)	25 (all layers) 10 (auxiliary) 15 (base layer)
Low	Enhancement (Auxiliary)	352 x 288/30	N/A	N/A	352 x 288/30
	Lower (Base)	-	N/A	N/A	352 x 288/30
	Bitrate (Mbits/s)	4 (all layers) 3 (base layer)	N/A	N/A	8 (all layers) 4 (auxiliary) 4 (base layer)

Fig. 9.3 – Perfis/Níveis da codificação MPEG-2 para modo escalonável e não-escalonável

Os sistemas HDTV americano e japonês utilizam o perfil *Main* e o nível *High*, escolha simbolizada por MP@HL. Na Europa, as combinações MP@H1440L e SSP@H1440L são algumas das combinações admitidas para HDTV.

Os perfis e níveis têm uma relação hierárquica. Assim, a sintaxe suportada por um par perfil/nível maior inclui todos os elementos sintáticos dos pares perfis/níveis menores, permitindo a um decodificador que trabalhe com um dado par, ser capaz de decodificar todos os *bitstreams* de pares menores que o seu.

A sintaxe do MPEG-2 tem duas categorias: 1) não-escalonável, que é um superconjunto da sintaxe de codificação do MPEG-1 com extensões adicionais que suportam sinais com entrelaçamento, e, 2) escalonável, que permite uma codificação do *bitstream* por camadas. Os decodificadores podem decodificar somente a camada básica, obtendo um sinal de qualidade mais baixa ou usar camadas adicionais para melhorar a qualidade do sinal decodificado [50].

9.3.1 - Descrição dos Processos de Codificação e Decodificação

Pode-se dizer que uma seqüência de vídeo consiste em uma coleção de quadros seqüenciais variantes no tempo. Cada quadro é uma matriz cujos pontos (*pixels*) representam uma imagem. As seqüências de vídeo tipicamente apresentam alta redundância temporal e espacial, assim, o padrão MPEG-2 utiliza o Método híbrido de Compressão, que para obter altas taxas de compressão realiza:

- Compressão temporal: através da estimação-compensação de movimento;
- Compressão espacial: através da Transformada DCT e codificação entrópica;

O MPEG-2 também explora a redundância psico-visual através da subamostragem da crominância. Os esquemas de um codificador e de um decodificador MPEG-2 são apresentados nas Figs. 9.4 e 9.5 [50], respectivamente. Antes de apresentar o esquema propriamente dito, serão colocadas algumas estruturas e conceitos básicos de forma a tornar mais clara a funcionalidade do sistema.

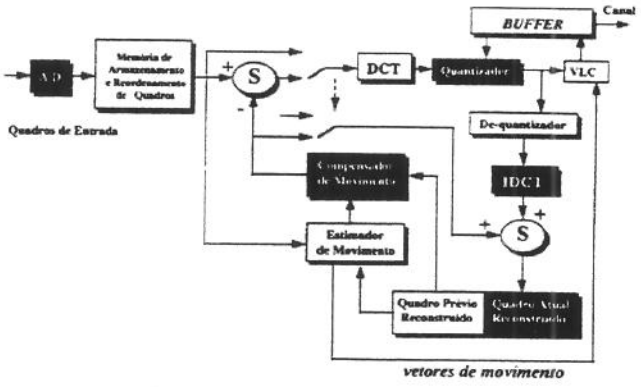


Fig. 9.4 – Codificador MPEG-2.

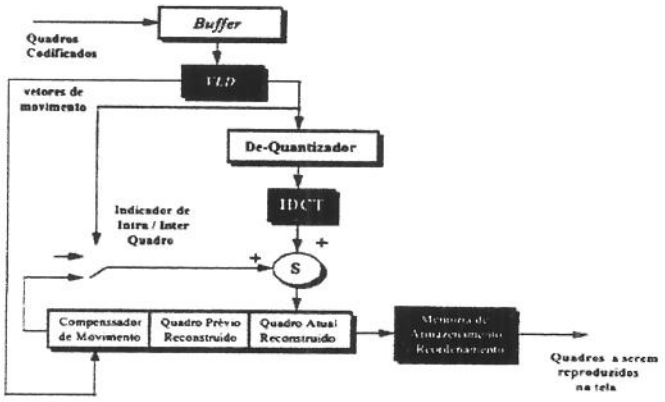


Fig. 9.5 – Decodificador MPEG-2.

9.3.2 - Partições da Seqüência de Vídeo

Os quadros componentes da seqüência de vídeo são agrupados em GOPs (*Group of Pictures*). O GOP é um conjunto de quadros que possibilita o acesso randômico. Usualmente, um GOP é composto de 6 a 15 quadros e é estabelecida uma ordem quanto aos tipos de quadros (tipo I, tipo P e tipo B). A ordem escolhida, comumente chamada de *estrutura* é repetida ao longo da seqüência, por exemplo, estrutura IBBP.

Cada quadro da seqüência de vídeo é dividido em unidades utilizadas no processo de codificação de forma a permitir a operacionalidade do sistema. Cada quadro é, portanto, segmentado em regiões não sobrepostas de *pixels*. A unidade básica de codificação na etapa de compressão temporal é o Macrobloco (MB) composto de 16x16 *pixels*. Na etapa de compressão espacial, cada MB é segmentado em regiões de 8x8 *pixels* chamadas "Blocos", que podem ser de luminância (Y) ou de crominância (Cr ou Cb).

Na Fig. 9.6 [53] é ilustrado um exemplo de MB de formato 4:2:0 (crominância subamostrada por um fator de dois, horizontal e verticalmente), logo, um MB composto de quatro blocos de luminância e dois de crominância. O MPEG-2 também suporta outros dois formatos: 4:2:2 (crominância subamostrada por um fator de dois somente na direção horizontal) e 4:4:4 (sem subamostragem da crominância).

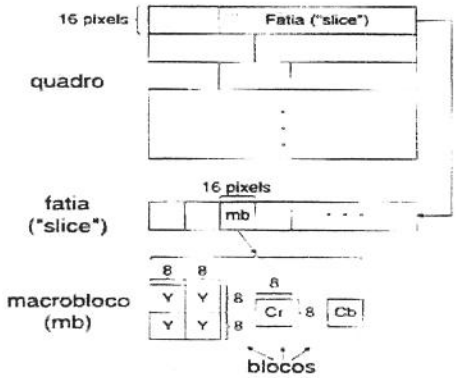


Fig. 9.6 – Estrutura de um quadro não-entrelaçado de formato 4:2:0.

Uma outra unidade de codificação é o *slice*, que consiste num conjunto horizontal de MBs (fatia de MBs), conforme também ilustrado na Fig. 9.6. Os *slices* servem como unidades de sincronismo. Podem existir áreas do quadro onde não há *slices*, portanto áreas que não são codificadas, conforme será visto mais adiante.

9.3.3 - Tipos de quadros

O MPEG-2 define três tipos de quadros de acordo com a compressão temporal utilizada

[51] [52]:

- Quadro-I (intracodificado): não sofre compressão temporal.
- Quadro-P (Predito): quadro cuja compressão temporal utiliza um quadro anterior como referência para a estimação-compensação de movimento.
- Quadro-B (Interpolado): quadros cuja compressão temporal utiliza dois quadros como referência para a estimação-compensação de movimento, sendo um quadro anterior e outro posterior.

Cada MB do quadro P ou B é codificado tomando-se a diferença entre o MB do quadro original (MB-original) e uma estimativa deste MB (MB-predição) gerada pela estimação-compensação de movimento. Nos quadros-P, esta estimativa é calculada tendo como referência um quadro anterior. Nos quadros-B, a estimativa final usada na codificação acima é a interpolação de duas estimativas iniciais: uma calculada tendo como referência um quadro anterior e outra calculada tendo como referência um quadro posterior. Neste processo, quadros do tipo B nunca são usados como referência uma vez que sofrem uma maior taxa de compressão, tendendo a apresentar uma maior distorção. Um exemplo de estrutura IBBP e a relação de predição entre os quadros do GOP encontram-se na Fig. 9.7 [50].

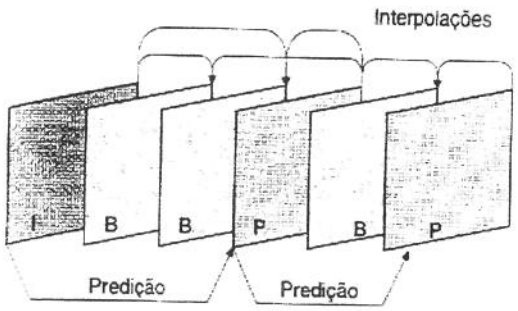


Fig. 9.7 – Exemplo de interdependência entre Quadros –I, –P e –B em uma seqüência de vídeo.

A ordem dos quadros codificados no *bitstream*, também chamada de ordem codificada (*coded order*) é a ordem na qual os quadros são codificados, transmitidos e decodificados. A ordem na qual os quadros reconstruídos são apresentados na etapa final do processo de decodificação, também chamada ordem de apresentação (*display order*), nem sempre é igual à ordem codificada, uma vez que pode ter sido feita a predição bidirecional (Quadros-B). Nesse caso, há a necessidade de uma reordenação dos quadros no decodificador.

9.4 - Estimação-Compensação de Movimento no MPEG-2

Para se explorar a correlação temporal da seqüência, utiliza-se técnicas de DPCM (*Differential Pulse Code Modulation*) para quadros tipo P e B com o objetivo, conforme já colocado, de codificar a diferença entre o MB original e sua predição ao invés de codificar o próprio MB original. O modo mais simples é considerar como MB-predição simplesmente o MB do quadro-referência que esteja nas mesmas coordenadas espaciais que as do MB-original. Trata-se do procedimento chamado de *Simple Frame Difference*. Contudo este procedimento só é eficiente em regiões da seqüência nas quais não há movimento [54].

Em regiões da seqüência que apresentem movimento, a correlação entre *pixels* de mesmas coordenadas espaciais é menor, logo, são adotadas técnicas de estimação-compensação de movimento para determinar o MB-predição. No MPEG-2, estas técnicas são baseadas no casamento de blocos (*Block Matching Criterion*). Para cada MB é realizada uma busca do vetor de movimento dentro de uma região do quadro-referência chamada janela de busca [-ρ, ρ], conforme ilustrado na Fig. 9.8. A busca para determinar o MB-predição deve ser restrita à janela de busca para simplificar a implementação de algoritmos e aumentar a velocidade da estimação de movimento, uma vez que este é o estágio computacionalmente mais extensivo de todo o processo de compressão.

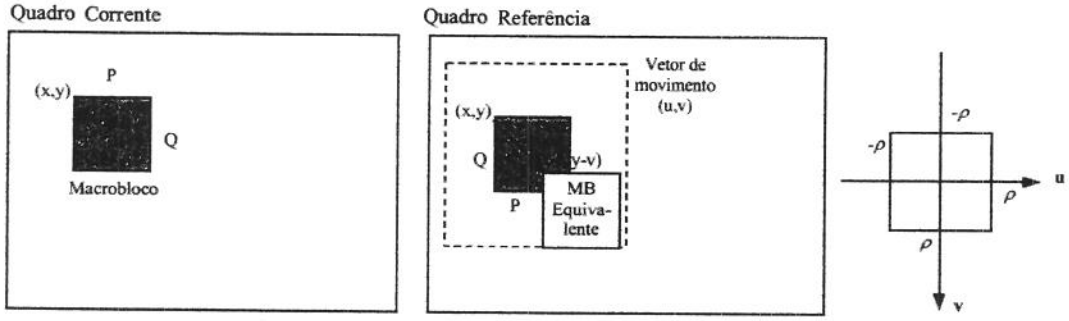


Fig. 9.8 – Representação esquemática da Estimação de Movimento.

Dentro da janela de busca, é escolhido como MB-predição aquele que minimizar uma medida de disparidade com relação ao MB-original. A função MAE (*Mean Absolute Error*) é a medida utilizada efetivamente pelos codificadores de vídeo práticos [50] e é dada por:

$$MAE(l,m)=\frac{1}{PQ}\sum_{i=0}^{P-1}\sum_{j=0}^{Q-1}\left|I\left(x_l+i,y_l+j,t\right)-I\left(x_l+i-l,y_l+j-m,t-k\right)\right| \qquad (9.1)$$

$$sendo\ -\rho \le l,m \le \rho$$

onde:

- $I(x_l + i, y_l + j, t)$ denota o MB original sendo que (x_l, y_l) são as coordenadas espaciais do *pixel* de seu canto superior esquerdo no quadro; (i, j) são as coordenadas dos *pixels* dentro deste MB; e *(t)* sua coordenada temporal na seqüência de vídeo.
- $I(x_l + i - l, y_l + j - m, t - k)$ analogamente denota um MB do quadro-referência sendo que (l, m) são os deslocamentos na janela de busca e *(t-k)* sua coordenada temporal.
- $P=Q=16$ são as dimensões do MB e $[-\rho,\rho]$ é a janela de busca.

A Fig. 9.8 ilustra um exemplo de estimação de movimento em que o deslocamento do MB atual em relação às mesmas coordenadas espaciais no quadro-referência é representado pelo vetor de movimento *(u,v)*. O padrão MPEG-2 utiliza ainda a precisão de meio-*pixel* (*half pixel accuracy*) para a determinação do vetor de movimento. Maiores detalhes podem ser encontrados em [43] [52].

Deve-se notar que as limitações da estimação de movimento residem no tamanho da janela de busca adotada (que deve ser maior para regiões da seqüência que apresentem movimento rápido) e no número de comparações feitas (cálculos de MAE). Fala-se do número de comparações feitas, uma vez que constam em literatura para estimação do MB-predição uma série de algoritmos rápidos que se aproveitam da característica direcional do movimento para somente calcular alguns valores de MAE dentro da janela de procura. Também há algoritmos subótimos que reduzem o número de pixels utilizados na comparação. Maiores detalhes podem ser encontrados em [43] [50].

9.5 - Determinação da Codificação dos MBs

O padrão MPEG-2 especifica exatamente a sintaxe do *bitstream* codificado e o processo de decodificação, porém permite várias opções para as operações do codificador. Foi visto que, de acordo com a forma de predição utilizada, um MB pode ser intracodificado, predito ou interpolado. O MPEG-2 contempla algumas variações destes três modos básicos, por exemplo, o modo *Dual Prime* e o modo *16x8 Motion Compensated*. Maiores detalhes podem ser encontrados em [51].

Os dados a serem codificados acerca de um MB predito ou interpolado são resultantes da diferença entre este MB-original e o MB-predição. Tem-se então um Macrobloco-diferença (MB-diferença) que a seguir será codificado de forma semelhante a um MB intracodificado, ou seja, passará pela compressão espacial (DCT e codificação entrópica). Os vetores de movimento são codificados por VLC (*Variable Length Coding*) diferencialmente com relação aos vetores de movimento decodificados previamente, com o objetivo de reduzir o número de bits necessário para

representá-los. A seguir, estes vetores de movimento são anexados ao *bitstream* codificado para serem transmitidos [51].

A divisão básica 1-3 quanto ao tipo de quadros também é otimizada pelo padrão. Nos quadros-P, além de MBs preditos, pode-se ter MBs intracodificados, o que equivale a utilizar um MB nulo como predição. Nos quadros-B, além de MBs interpolados, pode-se ter MBs intracodificados e até mesmo MBs preditos (em relação ao quadro-referência anterior ou posterior). A escolha de como será feita a predição para o MB fica a cargo do projetista do codificador e, geralmente, prioriza a maior compressão, dependendo assim da correlação entre o MB-original atual e o MB de melhor casamento dentro de cada quadro-referência. Todos os MBs de um quadro-I são intracodificados [53].

Dentro desta otimização, nos quadros P e B existe ainda a possibilidade de não se codificar todos os MBs. Neste caso, estes MBs são reconstruídos no decodificador utilizando-se somente os quadros-referência e considerando nulos os vetores de movimento, se o quadro atual é do tipo P, ou tomando-se os vetores de movimento do último MB codificado, se o quadro atual é do tipo B.

9.6 - Transformada Coseno Discreta (DCT)

O resultado das escolhas acima seja um MB-original ou um MB-diferença, é dividido em blocos 8x8 que sofrerão uma transformação DCT bidimensional. O objetivo desta transformação é promover a compactação de energia e descorrelação dos coeficientes, permitindo uma compressão mais eficiente por parte da codificação VLC subseqüente. O par da DCT-2D é definido por [54]:

transformada direta — *matricialmente*

$$C_{pq} = \alpha_p \cdot \alpha_q \sum_{j=0}^{N-1}\sum_{k=0}^{N-1} F_{jk} \cos\left[\frac{\pi(2j+1)p}{2N}\right]\cos\left[\frac{\pi(2k+1)q}{2N}\right] \qquad \boldsymbol{C} = \boldsymbol{T}^T \boldsymbol{F}\, \boldsymbol{T}$$

$$\text{onde } \alpha_0 \equiv \sqrt{\frac{1}{N}} \text{ e } \alpha_p \equiv \sqrt{\frac{2}{N}} \qquad 0 \le p,q \le N-1 \qquad \text{onde } T = \alpha_p \cos\left[\frac{\pi(2k+1)p}{2N}\right] \qquad 0 \le p,k \le N-1 \tag{9.6}$$

transformada inversa — *matricialmente*

$$F_{jk} = \sum_{p=0}^{N-1}\sum_{q=0}^{N-1} \alpha_p \cdot \alpha_q \cdot C_{pq} \cos\left[\frac{\pi(2j+1)p}{2N}\right]\cos\left[\frac{\pi(2k+1)q}{2N}\right] \qquad \boldsymbol{F} = \boldsymbol{T}^T \boldsymbol{C}\, \boldsymbol{T}$$

$$0 \le p,q \le N-1 \qquad \text{onde } \boldsymbol{T} = \alpha_p \cos\left[\frac{\pi(2k+1)p}{2N}\right] \tag{9.7}$$

$$\text{onde } \alpha_0 \equiv \sqrt{\frac{1}{N}} \text{ e } \alpha_p \equiv \sqrt{\frac{2}{N}} \quad p/\ 1 \le p \le N-1 \qquad 0 \le p,k \le N-1$$

onde F_{jk} é a intensidade dos *pixels* do bloco de dimensões NxN e C_{pq} são os coeficientes do bloco transformado.

9.7 - Quantização

Após a transformação DCT, os coeficientes resultantes são quantizados. O passo de quantização dos coeficientes é função das matrizes de quantização e de um parâmetro chamado *quantizer_scale*. Somente o coeficiente DC (freqüência (0,0)) de blocos intracodificados possui passo de quantização fixo, por ser o coeficiente que concentra a maior parte da energia do sinal. As matrizes de quantização recomendadas pelo padrão MPEG-2 para blocos intracodificados (blocos que sofreram somente compressão espacial) e para blocos intercodificados (blocos pertencentes a MBs que sofreram estimação-compensação de movimento) encontram-se na Fig. 9.9 [51]. O projetista do codificador pode utilizar outras matrizes que, neste caso, devem ser enviadas nos cabeçalhos (*headers*) dos dados codificados.

8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

(a)

16	17	18	19	20	21	22	23
17	18	19	20	21	22	23	24
18	19	20	21	22	23	24	25
19	20	21	22	23	24	26	27
20	21	22	23	25	26	27	28
21	22	23	24	26	27	28	30
22	23	24	26	27	28	30	31
23	24	25	27	28	30	31	33

(b)

Fig. 9.9 – Matrizes de Quantização recomendadas pelo padrão MPEG-2: (a) Matriz *Intra*; (b) Matriz *Inter*.

O parâmetro *quantizer_scale* é adaptativo a nível de MB e depende de um controle que é responsável por manter a taxa de bits constante na saída do codificador e promover o compromisso entre compressão e qualidade das imagens. O mecanismo específico pelo qual este parâmetro é alterado adaptativamente não é especificado pelo MPEG-2 [51]. Existem, porém, *Test Models* desenvolvidos para obter referências da qualidade de vídeo do padrão como o *Test Model 5* (TM5) que descrevem a alteração adaptativa do *quantizer_scale*. Uma descrição detalhada do MPEG-2 TM5 encontra-se em [52][55].

Os sistemas HDTV europeu e americano transmitem a taxas de bits constantes. O sistema americano [56] adotou taxas de 19,28 e 38,57 Mbps para transmissão terrestre e via cabo, respectivamente. Por outro lado, o sistema europeu terrestre [57] permite uma gama de taxas que

variam entre 4,98 e 31,67 Mbps, de acordo com parâmetros de modulação e de codificação de canal utilizados.

As formulações referentes à quantização não foram colocadas neste texto dada a sua diversidade de detalhes para coeficientes DCs e ACs dos diferentes tipos de MBs. Em [54] encontra-se um estudo detalhado a esse respeito baseado no manual do MPEG-2 [51].

9.8 - Codificação entrópica

Após a quantização, o bloco é preparado para sofrer codificação entrópica utilizando uma varredura *zig-zag*, ilustrada na Fig. 9.10. Antes da codificação entrópica, o coeficiente DC é subtraído do valor do coeficiente DC do bloco anterior, uma vez que usualmente há uma forte correlação entre os coeficientes DC de blocos vizinhos. É essa diferença DPCM que será codificada entropicamente no caso do DC.

Os demais coeficientes, chamados ACs, sofrem a varredura *zig-zag* antes da codificação entrópica. Isto porque a compactação de energia da DCT e subseqüente quantização fazem com que muitos coeficientes ACs sejam zerados. A varredura *zig-zag* desses coeficientes faz com que se formem grandes trechos com valores nulos, tornando a codificação RL subseqüente muito eficiente.

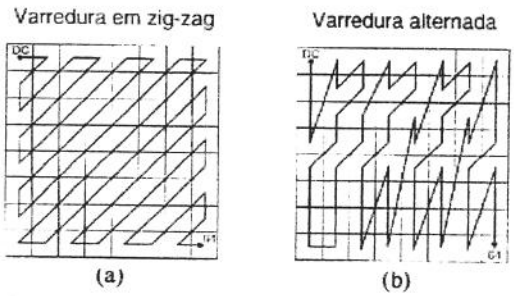


Fig. 9.10 – Modos de Varredura dos coeficientes do bloco transformado.

A varredura alternada, apresentada na Fig. 9.10b, é utilizada quando a seqüência de vídeo é entrelaçada e o bloco codificado contém informações dos campos par e ímpar.

Após a varredura é realizada uma variante da codificação de *Huffman*. Fazer uma tabela de *Huffman* para toda a gama de valores dos coeficientes quantizados seria inviável. Para contornar este problema, os valores são classificados primeiramente em níveis (*levels*) de acordo com sua amplitude. Após o DPCM, o valor DC é codificado utilizando o conceito de níveis num par (L/A). L é o *level* obtido através da Tabela 9.1 e codificado através das Tabelas B.12 e B.13 que constam em [51]. *A* é o modulo do resultado do DPCM em binário com *L* bits e substituindo-se o bit mais significativo por um bit de sinal.

Já os DCs de blocos intracodificados e os componentes DC e ACs de blocos intercodificados são codificados de forma diferente. Os blocos após a varredura *zig-zag* são pré-codificados através de um sistema chamado *Run-Level* (*R/L*). *R* indica o número de ACs nulos desde o último coeficiente não-nulo; e *L* é o *Level* segundo a Tabela 9.1. Ao contrário dos blocos intra, nos blocos inter o DC também entra no cálculo do *Run*. Após o último valor não nulo do bloco, envia-se o código EOB (*End of Block*), indicando que a partir dali todos os coeficientes são nulos. O par *R/L* é então codificado por uma tabela fixa (Tabelas B.14 a B.16, que constam em [51] e é seguido pela amplitude A. A é o modulo da amplitude do coeficiente em binário com *L* bits e substituindo-se o bit mais significativo por um bit de sinal.

Tabela 9.1 – Relação entre *Level* e *Amplitude* do coeficiente para DC

Level	**Amplitude (em módulo)**
0	0
1	1
2	2,3
3	4 a 7
4	8 a 15
5	16 a 31
6	32 a 63
7	64 a 127
8	128 a 255
9	256 a 511
10	512 a 1023
11	1024 a 2047
12	2048 a 4095
.	.
.	.
.	.
40	2^{39} à 2^{40}-1

Por exemplo, seja o DC do bloco anterior igual a 128 e o resultado da varredura *zig-zag* de um bloco intracodificado dada por:

160, -6, 1, 0, -1, 1, -1, 0, 0, 1, 0, 0, 0, ...0.

Aplicando a codificação descrita acima, pode-se mapear este bloco como a seqüência de símbolos: Aplicando a codificação *Run-Level* e utilizando a Tabela 9.1, pode-se mapear esse bloco como a seqüência de símbolos:

(6,32) (0/-6) (0/1) (1/-1) (0/1) (0/-1) (2/1) EOB

Dessa forma, utilizando as Tabelas B.12 e B.14 constantes em [51], esse bloco seria codificado como:

$$\underbrace{\underbrace{11110}_{(L)}\ \underbrace{100001}_{sA}}_{11\,bits};\underbrace{\underbrace{00100001}_{(R/L)}\ \underbrace{1}_{s}}_{9\,bits};\underbrace{\underbrace{11}_{(R/L)}\ \underbrace{0}_{s}}_{3\,bits};\underbrace{\underbrace{011}_{(R/L)}\ \underbrace{1}_{s}}_{4\,bits};\underbrace{\underbrace{11}_{(R/L)}\ \underbrace{0}_{s}}_{3\,bits};\underbrace{\underbrace{11}_{(R/L)}\ \underbrace{1}_{s}}_{3\,bits};\underbrace{\underbrace{0101}_{(R/L)}\ \underbrace{0}_{s}}_{5\,bits};\underbrace{\underbrace{10}_{EOB}}_{2\,bits} \Rightarrow 40\,bits$$

onde:

L = *Level* Codificado VLC através da tabela B.12 de [51];

R/L = *Run-Level* Codificado VLC através da tabela B.14 de [51];

A = Amplitude do módulo do Componente;

s= bit de sinal: 0 para positivo e 1 para negativo;

Assim, para codificar completamente esse bloco, o MPEG-2 utiliza apenas 43 bits, o que representa uma taxa de compressão de aproximadamente *64* x *8/40* = *12,8:1*.

As tabelas de códigos VLC são pré-definidas pelo padrão e não podem ser modificadas.

9.8.1 - Atualização de Quadros Referência

Um dos problemas decorrentes do uso da codificação preditiva é a propagação de erros. Por exemplo, se ocorre erro na decodificação de um MB que servirá de referência para codificar outros MBs, o erro se propaga. Para contornar o problema, o padrão MPEG-2 recomenda que cada MB que ocupa uma determinada posição espacial deve ser intracodificado no máximo a cada 132 quadros. Para os sistemas de HDTV recomenda-se a codificação de quadros intracodificados em intervalos de no máximo 0,5s [53]. Para HDTV recomenda-se também que seja enviado o cabeçalho da seqüência antes de cada quadro intracodificado. Estes procedimentos se fazem necessários uma vez que quando o usuário sintoniza um determinado canal, essas informações de cabeçalho são necessárias para o início do processo de decodificação e reprodução do sinal de vídeo.

9.9 - Conclusões

Neste capítulo foram apresentados conceitos básicos de codificação de vídeo e um breve resumo do sistema de funcionamento do padrão de compressão de vídeo MPEG-2. Foram abordados os conceitos de vídeo digital, codificação com e sem perdas, entrelaçamento, tipos de quadros, compressão temporal e espacial. No próximo capítulo será apresentado o algoritmo de estimação de movimento multi-resolução no domínio da transformada wavelet.

Capítulo 10
Algoritmos de Estimação de Movimento em Domínio Wavelet

10.1 - Introdução

Apesar dos conhecidos bons resultados com a TW para imagens estáticas [59][60][61], aplicações desta transformada na compressão de vídeo [40][62][63][64] são recentes e ainda pouco exploradas, abrindo um vasto campo para pesquisa.

Como foi visto no Capítulo 9, os atuais algoritmos de compressão de vídeo possuem uma estrutura híbrida (temporal e espacial). Segundo [63], para a codificação de vídeo, há três formas conceitualmente diferentes para usar a TW reportadas até agora na literatura:

1. Utilizar a TW 3D;
2. TW dos quadros originais, seguida de estimação-compensação de movimento dos coeficientes da wavelet;
3. Estimação-compensação de movimento tradicional no domínio do tempo, seguido da aplicação da TW no quadro-erro compensado.

Como na presente pesquisa a base do sistema de compressão de vídeo desenvolvido é um sistema de compressão de imagens, optou-se pela adoção do segundo método: fazer-se estimação de movimento em domínio wavelet para aproveitar melhor os bons resultados adquiridos na compressão de imagens.

Nesse método, a compressão de vídeo baseada na TW proporciona uma descorrelação espacial eficiente e o algoritmo de estimação de movimentos para esta transformada possui baixa complexidade [63], uma vez que se pode explorar a grande correlação entre as sub-bandas. A Fig. 10.1 mostra o diagrama de blocos de um codificador híbrido baseado na TW [26] com estimação de movimento em domínio *wavelet*.

Seguindo esse conceito, Zhang *et. al.* [4] propuseram uma técnica de estimação multi-resolução (MRME - *Multi Resolution Motion Estimation*) que estima os vetores de movimento (MVs) no domínio da TW de acordo com o índice da camada de decomposição. Desta forma, os MVs das subbandas de detalhes e a subbanda de baixas freqüências da última camada de

decomposição são estimados via um algoritmo tradicional, semelhante aos utilizados no MPEG-2 [51]. OS MVs das demais camadas são calculados através de uma predição que utiliza os vetores da próxima camada como referência. Uma descrição mais detalhada será apresentada na seção 10.2.

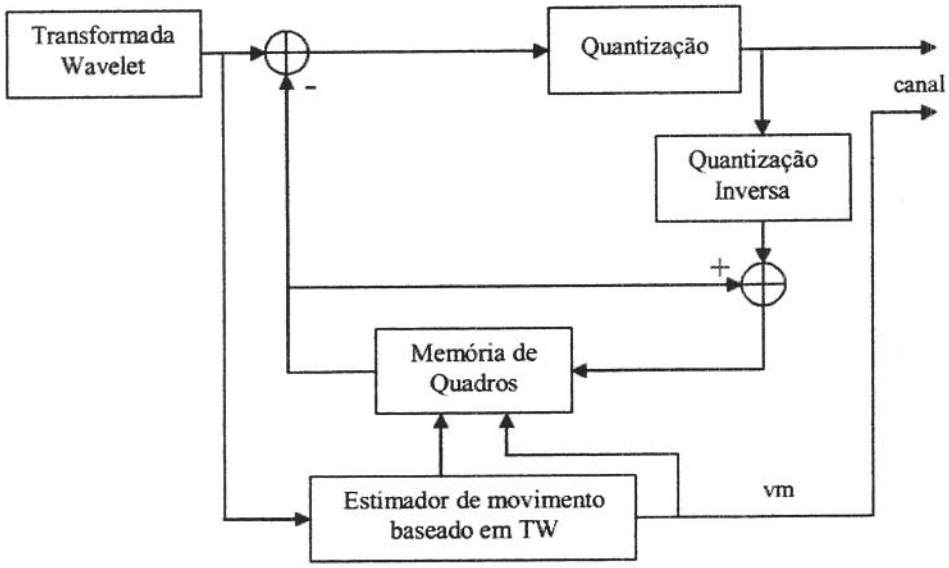


Fig. 10.1 - Codificador de Vídeo Híbrido baseado na TW

Outros esquemas bastante interessantes que trabalham com estimação-compensação de movimento conjuntamente com *wavelets* encontram-se em [64] [65] [66].

Quanto à quantização, como cada sub-banda da TW possui características próprias, melhores resultados podem ser obtidos se cada sub-banda for quantizada de forma independente [67]. Além disso, a forte relação da TW com o mecanismo de visão humana permite que componentes da imagem de menor importância visual sejam identificados e eliminados mais facilmente. Obviamente quanto maior o descarte, maior a compressão e degradação da imagem resultante.

Além disso, a TW provê uma maneira fácil e direta de se implementar a codificação por escalonabilidade espacial. O objetivo dos codificadores de vídeo que adotam escalonabilidade é oferecer uma compatibilidade/interoperabilidade entre diferentes serviços ou padrões, o que poderá servir, por exemplo, para a transmissão de apenas um sinal de televisão digital para aparelhos receptores com diversas resoluções, evitando a utilização de canais diferentes para cada aplicação. Um exemplo de codificação de vídeo com escalonabilidade espacial baseado na TW está na Fig. 10.2, construído através de algumas modificações no esquema da Fig. 10.1 [26].

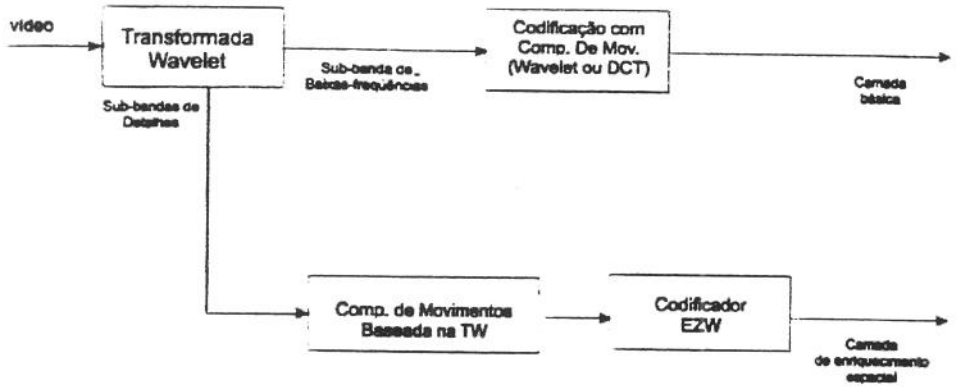


Fig. 10.2 - codificador com escalonabilidade espacial baseado na TW

10.2 - MRME – *Multi-Resolution Motion Estimation*:

Usualmente, em qualquer sistema de codificação de vídeo a etapa de maior complexidade computacional é a estimação de movimento. Isso ocorre porque o processo de casamento (*matching*) que é usado para a determinação dos vetores de movimento é extremamente exaustivo.

No caso da TW, Zhang *et. al.* [4] propôs um algoritmo relativamente simples e que apresenta uma complexidade computacional reduzida em conjunto com uma boa estimação do movimento (baixo erro de predição). A eficiência desse algoritmo vem da exploração da semelhança entre as estruturas das sub-bandas, que é uma característica das imagens transformadas por *wavelet*. Posteriormente Swammy *et. al.* [5] desenvolveu e testou algumas modificações desse algoritmo obtendo resultados ainda melhores. Como os algoritmos de Swammy *et. al.* englobam o conceito original, apresentar-se-á aqui apenas essas variantes.

10.2.1 -Princípio de funcionamento

Foi visto que a TW decompõe um *frame* de vídeo num conjunto de sub-imagens com diferentes resoluções correspondentes a diferentes bandas de freqüências. Estas sub-imagens multi-resolução fornecem uma representação da estrutura do movimento global do sinal de vídeo em diferentes escalas. As atividades de movimento de uma sub-imagem particular em diferentes resoluções são diferentes, mas altamente correlatas uma vez que, na verdade, especificam a mesma estrutura de movimento em diferentes escalas.

Num esquema MRME, o campo de movimento (formado pelos Vetores de Movimento – MVs) é primeiramente calculado para as sub-imagens de resolução mais baixa, situada no canto

superior direito Fig. 10.3. A seguir, os MVs nas resoluções mais altas são preditos pelos MVs nas resoluções mais baixas e são refinados a cada passo.

Este esquema reduz consideravelmente o tempo de procura e casamento dos blocos (reduzindo a complexidade computacional da Estimação de Movimento); fornece uma caracterização completa da estrutura de movimento intrínseca; produz um campo vetorial de movimento suave; e evita as falhas do MRMC com tamanho de bloco constante em descrever o movimento de pequenos objetos.

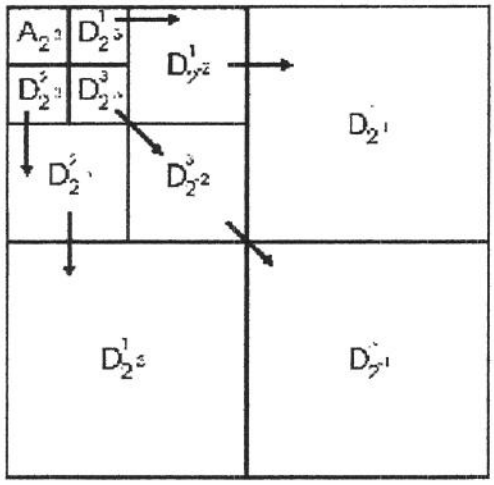


Fig. 10.3 – Estrutura de imagem transformada pela TW.

Swammy *et. al.* trabalhou com 3 níveis de resolução neste codificador, como a mostrada na Fig. 10.3, variando o tamanho do bloco de acordo com a resolução de cada nível. Assim, para M níveis de resolução, foi utilizado um tamanho de bloco de $p \cdot 2^{M-j} \times q \cdot 2^{M-j}$ para a sub-imagem do j-ésimo nível, i.e., quanto menor a resolução, menor o tamanho do bloco. As constantes p x q indicam o tamanho do bloco na resolução mais baixa, em geral 2x2.

Com esta estrutura, todas as sub-imagens escaladas tem o mesmo número de blocos de movimento. A aproximação com blocos de tamanho variável "pesa" a importância dos diferentes níveis e casa com a percepção visual humana para diferentes freqüências a diferentes resoluções. Supera a estimação de movimento (ME) tradicional por ser capaz de detectar o movimento de pequenos objetos no nível mais alto da pirâmide e supera a compensação de movimento (MC) tradicional ao requerer um número menor de cálculos (uma vez que nenhuma interpolação é necessária a medida em que a grade vai sendo refinada).

Além destas vantagens, a aproximação dos autores produz sub-imagens residuais com baixíssima energia, resultando num processo "limpo" para a propagação da estimação do movimento nos níveis mais baixos.

10.2.2 - Introdução da notação utilizada

Para introduzir notações, se faz necessário um detalhamento um pouco maior dos processos de estimação/compensação de movimento introduzidos anteriormente. Na ME, a imagem é particionada em blocos, e o mesmo deslocamento (MV) é assumido para todos os *pixels* dentro de cada bloco. Neste caso, no modelo de movimento, assume-se que a imagem é composta por objetos rígidos de movimento translacional. Embora este modelo seja restritivo, é justificado pelo fato de que o movimento complexo (rotacional) pode ser representado por uma soma de componentes translacionais.

Seja o bloco no frame t referenciado por seu *pixel* superior esquerdo $I_t(s)$ localizado na posição $s=(s_x,s_y)$. Seja $v(x,y)$ o vetor de movimento representando o movimento do bloco entre dois *frames* consecutivos $t-1$ e t. Conforme mostrado na Fig. 10.4, é possível encontrar através de uma procura, o melhor bloco correspondente para $I_t(s)$ no frame anterior $t-1$ e assim determinar v. Este é o princípio básico da estimação de movimento.

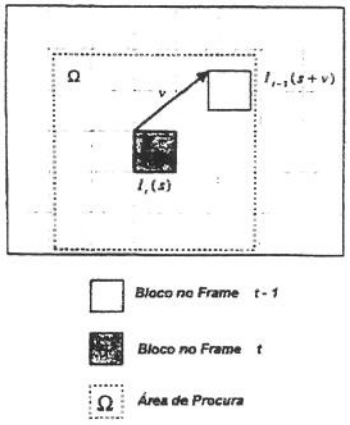


Fig. 10.4 – Área de procura da estimação de movimento.

Nos algoritmos de *block-matching* o *frame* é dividido em blocos e então, para cada bloco no frame-corrente no tempo t, é procurado um bloco correspondente no *frame* anterior ($t-1$) dentro de uma janela de procura Ω. É considerado como correspondente o bloco que segundo um critério de comparação com o bloco-atual minimize uma certa função de diferenças. O critério de casamento de blocos pode ser qualquer métrica, mas é usual se escolher a métrica MAD (*minimum absolute*

difference) por causa da simplicidade de cálculo. Usando essa métrica, para obter-se o vetor de movimento de um bloco $I_t(s)$ usa-se a expressão:

$$v = \arg\min_{v \in \Omega} \sum_{s \in W} \| I_t(s) - I_{t-1}(s+v) \| \tag{10.1}$$

Encontrar o melhor bloco-correspondente só pode ser garantido se for feita uma procura completa (*Full-Search*) da série de deslocamentos-candidatos dentro da área de procura Ω no frame-referência. À despeito de ser computacionalmente extensivo, o *Full-Search* é o mais usado em virtude de sua simplicidade e facilidade de implementação.

Outro fator que influi no desempenho da ME é o tamanho de Ω. Uma pequena área de procura pode facilmente estimar pequenos movimentos de objetos e possui uma baixa complexidade computacional. Já para movimentos rápidos, requer-se uma área de procura maior, elevando muito a complexidade.

Determinado o vetor de movimento, o bloco no frame-referência deslocado por $v(x,y)$ gera um bloco-predição que é subtraído do bloco no frame-corrente. A este processo dá-se o nome de Compensação de Movimento, gerando o que se chama de Sinal-residual do bloco.

10.3 - O algoritmo MRME:

O algoritmo MRME explora a correlação cruzada entre os níveis de resolução *wavelet* com o objetivo de reduzir a complexidade computacional do processo de estimação de movimento (ME) na TW. Um exemplo da MRME proposta está ilustrado na Fig. 10.5.

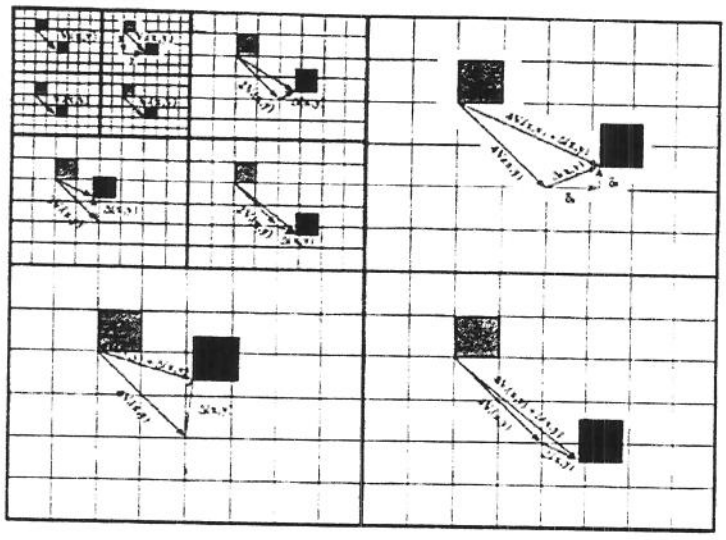


Fig. 10.5 – MRME com Tamanho de Bloco Variável

Dentro deste esquema, a estimação de movimento completa é feita em todas as sub-imagens do nível *3* usando blocos de *2x2*. As predições dos VMs dos níveis de resolução mais altos é feita a partir da sub-imagem do mesmo tipo no nível *3* (Vertical, Horizontal ou Diagonal) multiplicando-se por 2^{3-j}, onde $j \leq 3$ representa o nível de resolução. Essas predições são refinadas através de uma procura dentro de cada sub-imagem usando uma região de procura reduzida.

Matematicamente, pode-se expressar esse método por:

$$V_j^i(x,y) = 2V_{j+1}^i(x,y) + \Delta_j^{'i}(\delta_x,\delta_y) \quad \text{para } i = 1,2,3 \text{ e } j = 1,2 \tag{10.2}$$

Onde: $V_j^i(x,y)$ são os MVs centrados em *(x,y)* para a sub-imagem $D_{2^{-j}}^i$, o subescrito j indica o nível e o sobrescrito i indica a sub-imagem dentro do nível.

Esta configuração produz energias muito baixas em todas as sub-imagens residuais, porém, apresenta comparativamente um número maior de cálculos e um gera um maior gasto em bits pela necessidade da transmissão dos refinamentos Δ.

Certamente este esquema pode ser simplificado usando MVs independentes obtidos para $A_{2^{-3}}$ como base inicial (para todas as sub-imagens) e então escalados e refinados para todas as sub-imagens de níveis mais baixos usando o *Full-Search* com uma área menor de procura. Com este esquema, após o escalonamento de $V_3^0(x,y)$; o bloco no *frame* anterior é primeiro deslocado por $V_j^i(x,y)$ e daí é feita a procura de movimento para achar $\Delta'(\delta_x,\delta_y)$, o que equivale a achar:

$$\Delta'(\delta_x,\delta_y) = \underset{\delta x,\delta y \in \Omega'}{arg\,Min}\left[\frac{1}{XY}\sum_{k=-X/2}^{X/2}\sum_{l=-Y/2}^{Y/2}\left|I_t(s_x+k,s_y+l) - I_{t-1}(s_x+k+x+\delta x, s_y+l+y+\delta y)\right|\right] \tag{10.3}$$

Os MVs no nível j são:

$$V_j^i(x,y) = V_3^0(x,y)\cdot 2^{M-j} + \Delta_j^{'i}(\delta_x,\delta_y) \quad para\ \{i = 1,2,3 \text{ e } j = 1,2,3\} \text{ e } j < M \tag{10.4}$$

Onde $V_3^0(x,y)$ é o MV para a sub-imagem $A_{2^{-3}}$ e $\Delta'(\delta_x,\delta_y)$ é o incremento do MV encontrado pelo *Full-Search* com área de procura reduzida. O tamanho de Ω' é o mesmo de Ω, mas como o tamanho do bloco é aumentado por um fator de *2* em ambas as dimensões; a área de procura efetiva é reduzida.

Os cálculos e o *overhead* de movimento podem ser reduzidos drasticamente se os MVs para cada sub-imagem não forem refinados. É importante salientar o fato de que estes refinamentos não

apresentam precisão *subpixel*, mas sim um refinamento para chegar à precisão de 1 pixel em cama camada. Este esquema pode ser visto na Fig. 10.5 zerando todos os $\Delta'(\delta_x, \delta_y)$.

O sistema proposto foi implementado em tempo real e foram testadas as sequências: "*Car*" 720x480, 16 bits/pixel, entrelaçada, à cores e com grande movimento rápido, ideal para testes com esquemas MC; "*Cheeleaders*" e "*Football*". A Tabela 10.1 mostra a distribuição de energia entre diferentes sub-imagens residuais antes e após aplicar o MRMC com tamanho de bloco variável para um típico frame de vídeo na seqüência "CAR".

Tabela 10.1 – Distribuição de energia entre as sub-imagens residuais para um frame típico da seqüência *Car*

Energia	A_8	D_8^1	D_8^2	D_8^3	D_4^1	D_4^2	D_4^3	D_2^1	D_2^2	D_2^3
Original signal	49587.23	7361.20	452.91	148.47	1391.86	65.89	18.46	203.53	7.48	3.31
A_8 only	336.00	848.63	155.74	193.20	428.40	53.07	37.12	110.65	5.26	1.28
A_8, D_8^i only	336.00	195.58	44.32	28.34	414.99	44.54	21.62	97.47	4.31	0.55
A_8 + refine	336.00	186.16	44.56	29.01	138.83	13.58	4.65	39.96	1.02	0.05
A_8, D_8^i + refine	336.00	195.58	44.32	28.34	139.71	15.14	7.37	38.61	0.78	0.19

A notação de A_8, D_8^1, D_8^2, D_8^3, D_4^1, ... , D_2^3 usada pelos autores se refere à sub-imagens $A_{2^{-3}}$, $D^1_{2^{-3}}$, $D^2_{2^{-3}}$ e $D^3_{2^{-3}}$, $D^1_{2^{-2}}$, ..., $D^3_{2^{-1}}$ respectivamente. As *4* variantes do MRME descritas na tabela correspondem à:

"A_8 only": *Full-Search* somente em A_8 e todas a outras sub-imagens usam os mesmos VMs (ou VMs escalados) de A_8 (Δ=*0* na Fig. 10.5);

"A_8, D_8^i only": *Full-Search* somente em A_8 e em D_8^i (nível 3). A informação de movimento é então propagada direcionalmente para as outras sub-imagens sem refinamento (sem o segundo *Full-Search*) (Fig. 10.5).

"A_8+refine": *Full-Search* somente em A_8. Os VMs para as outras sub-imagens são preditos a partir dos VMs de A_8 (Fig. 10.4), fazendo um segundo *Full-Search*.

"A_8, D_8^i +refine" : é uma extensão da Fig. 10.5. Os VMs para as outras sub-imagens são preditos via um segundo *Full-Search* , tendo como ponto de partida os VMs de A_8 e D_8^i.

As médias de todos os sinais-residuais são muito baixas (0,01 a 3,75), entretanto, a variância depende da atividade do movimento e da precisão da estimação. Pode-se ver da Tabela 10.1 que a decomposição compacta a maior parte da energia em A_8. Após o MRMC com blocos de tamanho variável as energias ou variâncias na maioria das sub-imagens são consideravelmente reduzidas.

A redução nos níveis mais altos (especialmente em A_8) é ainda mais significativa devido ao esquema de estimação de movimento altamente preciso usado para este nível e por causa das variações entre *frames* sucessivos serem bem compensadas. Pode ser facilmente observado que *"A_8+refine"* e *"A_8, D_8^i +refine"* produzem energias menores do que os outros dois esquemas.

10.4 - Conclusões

Neste capítulo foi apresentadas as variantes propostas por Swammy *et. al.* do algoritmo de estimação e compensação de movimento em domínio wavelet denominado Estimação de Movimento Multi-Resolução (MRME) desenvolvido originalmente por Zang *et. al.* Neste algoritmo, cada sub-imagem é dividida em blocos quadrados proporcionais ao sua resolução de forma que para cada bloco seja associado um vetor de movimento. A economia na transmissão dos vetores de movimento extras consiste na utilização da direcionalidade espacial da representação wavelet para prever os vetores de movimento das sub-imagens de maior resolução a partir das de menor resolução.

No próximo capítulo serão apresentados os codificadores de imagens estáticas e de vídeo desenvolvidos durante a pesquisa, e os respectivos parâmetros de simulação.

Capítulo 11
Codificadores Propostos

11.1 - Introdução

Como dito anteriormente, uma das maiores desvantagens dos esquemas de compressão fractal de imagens é o tempo de codificação. Este atraso é devido, em sua maior parte, à alta complexidade computacional da etapa de procura do casamento entre bloco-imagem e bloco-domínio. Para aliviar este atraso foram desenvolvidas técnicas de pré-classificação dos blocos-domínio, como as mostradas no Capítulo 4, de forma a reduzir o número de comparações realizadas para cada bloco-imagem.

Mesmo com a utilização destes métodos de classificação, o tempo de codificação do método fractal permanece muito acima do tempo necessário para possibilitar implementações em codificação de vídeo. Por exemplo, pode-se citar que a implementação feita em Matlab do algoritmo de Fisher, aplicada a algumas imagens *512x512 pixels*, 8bpp, *overlap* de *50%*, demorou, em média, cerca de 30 minutos para completar a codificação.

Por outro lado, a codificação por transformada wavelet discreta (TWD) é uma técnica bem desenvolvida, com inúmeras publicações na literatura, e que apresenta um sistema de codificação rápido e eficiente, mas que apresenta algum efeito de *ringing* nas bordas naturais da imagem.

A proximidade entre as técnicas de codificação fractal e transformada wavelet é que, sob diferentes aspectos, ambas as técnicas procuram explorar a correlação, ou a similaridade, entre a informação em diferentes escalas (níveis de resolução) da imagem a ser codificada.

A partir desta análise, foi desenvolvida uma proposta de aceleração da codificação fractal de imagens utilizando a TWD e, posteriormente, um codificador fractal-*wavelet* de vídeo digital. Nas Seções 8.2 e 8.3 serão apresentados, respectivamente, o codificador de imagens e o codificador de vídeo.

11.2 - Codificador Fractal-Wavelet de Imagens

Analisando o algoritmo de codificação fractal fica claro que o maior responsável pela grande tempo de codificação é o algoritmo de procura. Para tentar buscar um método para reduzir a complexidade deste algoritmo foi feita uma análise da complexidade do mesmo.

11.2.1 - Análise da complexidade do algoritmo de procura

Analisando mais profundamente o algoritmo de procura do codificador fractal para uma imagem *NxN*, pode-se calcular que o número de comparações *C* realizadas para cada bloco-imagem do nível de quadtree *n* é, para um *overlap* máximo (blocos domínio alinhados em uma grade com passo de 1 *pixel*) dado por

$$C = N^2\left(\frac{2^n - 1}{2^n}\right) \tag{11.1}$$

Multiplicando este valor pelo número de blocos-imagem $(N/2^n)^2$ tem-se que a complexidade do algoritmo de procura é $O[N^4]$. Utilizando o processo de classificação dos blocos, o número de comparações médio é obtido dividindo o número de comparações sem classificação pelo número de classes existentes o que reduz a quantidade de comparações, mas não a ordem da complexidade.

Desta forma para acelerar o processo de codificação a idéia principal é tentar reduzir o tamanho do *domain pool* efetivo para cada bloco-imagem. Para alcançar esta redução pode-se alterar o algoritmo em vários pontos, como a redução do *overlap*, a utilização de classificações com mais classes (mais restritivas) e a diminuição do tamanho da imagem. No entanto, a redução do domain-pool acaba gerando uma limitação na procura por auto-similaridade, o que causará uma perda na qualidade da imagem reconstruída.

11.2.2 - A proposta

Neste ponto da análise, observou-se que há uma forma de reduzir o tamanho da imagem sem acarretar a perda de detalhes utilizando a TWD, mostrada nos Capítulos 5 a 8. Resumidamente, a TWD transformada realiza a cada aplicação (nível) a decomposição da imagem em quatro subimagens distintas, cada uma com metade da resolução da imagem original: uma subimagem Aproximação (versão passa-baixas da imagem original) e três subimagens de Detalhes (passa-altas somente na vertical, somente na horizontal, e em ambas as direções). Aplicando-se recursivamente a TWD sobre as aproximações pode-se reduzir o tamanho da imagem por um fator de 2^{Nd}, onde *Nd* é o número de níveis de decomposição.

Desta forma, ao aplicar o algoritmo fractal à subimagem de Aproximação, o tempo de codificação desta imagem fica significativamente reduzido e os detalhes que seriam perdidos em um caso geral, ficam armazenados nas subimagens de detalhes que ainda podem ser codificados e enviados melhorando assim a qualidade da imagem recuperada.

Para codificar as sub-imagens de detalhes, em nossa proposta foi utilizada uma versão ligeiramente modificada do algoritmo SPIHT (*Set Partitioned in Hierarchical Trees*).

A Fig. 11.1 abaixo apresenta o esquema básico do codificador proposto.

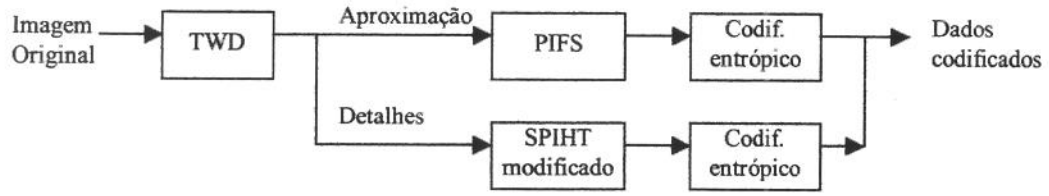


Fig. 11.1 – Esquema do codificador Híbrido Fractal-*Wavelet*

Para que o algoritmo SPITH se encaixasse no esquema híbrido proposto foi modificada a inicialização da LIP que originalmente conteria todos os elementos subimagem de Aproximação. No entanto, o algoritmo híbrido codifica esta subimagem usando fractal, então apenas foram retirados todos os elementos da LIP e ela foi inicializada vazia.

A LSP e a LIS não foram alteradas, pois a LSP já é originalmente vazia e a LIS contém apenas a descendência dos coeficientes da subimagem aproximação (que estão nas subimagens de detalhes). Para o cálculo do *n0* e dos *threshold* inicial somente foram considerados os coeficientes que serão efetivamente codificados pelo SPIHT, ou seja, somente os valores das subimagens de Detalhes.

Como o algoritmo SPIHT permite que seja especificado um valor de taxa de bits almejado, o método híbrido realiza primeiramente a codificação fractal e estima o número de bits usado pelos dados (pela entropia de primeira ordem) e passa à etapa do SPIHT modificado a taxa remanescente de forma que o *bitstream* final apresente a taxa desejada.

O último detalhe da implementação deste algoritmo é que foi realizada uma pequena modificação no algoritmo fractal básico de fidelidade almejada (Fig. 3.4). Como reduzimos significativamente o tamanho da imagem que será codificada por fractal, para evitar que a redução do domain-pool acarretasse em erros excessivos na subimagem de Aproximação, se a fidelidade mínima não for atingida no último nível de *quadtree* permitido, o bloco é marcado como não codificado e seus coeficientes são enviados com precisão de 8 bits por *pixel*.

Esta modificação foi adotada porque a TW concentra grande parte da energia da imagem na subimagem de Aproximação, e admitir perdas muito grandes na codificação desta subimagem acarreta grande perda de qualidade na imagem reconstruída. Para reduzir o *overhead* desta modificação, utilizamos o parâmetro de máximo nível de *quadtree* permitido de forma que o menor bloco seja sempre de tamanho 2x2 *pixels*.

11.2.3 - Parâmetros de simulação

1) TWD e SPIHT:

- Wavelet usada: Spline Biortogonal 9-7;
- Níveis de decomposição: 3 níveis;

Tanto para o codificador SPIHT quanto para o Híbrido foi utilizada uma decomposição *wavelet* de 3 níveis usando a *wavelet* spline biortogonal 9-7 [25] (comumente usada nos artigos de codificação de imagem).

O algoritmo SPITH possui apenas um parâmetro de codificação que é a taxa de bits almejada que foi ajustada de acordo.

2) Fractal:

Foram usados os seguintes parâmetros:

- Nível máximo de Quadtree: 7;
- Nível mínimo de Quadtree: 3;
- *Overlap*: 50%;
- Número de bits para S_i e O_i: 5 e 7 respectivamente;

A taxa de bits foi ajustada através do parâmetro de fidelidade almejada (Fig. 3.4).

3) Híbrido:

Foram usados os mesmos parâmetros que os utilizados nos algoritmos individuais com exceção do nível máximo de *quadtree* que teve que ser reduzido de forma que o menor bloco a ser processado em cada caso tivesse 2x2 *pixels*.

11.3 - O codificador proposto aplicado a vídeo

O problema de exaustivo tempo de processamento fractal é potencializado quando se trata da codificação de vídeo, que usualmente é adotada para aplicações em tempo real. Este é o motivo pelo qual não se encontram na literatura científica, nem em uso prático, codificadores de vídeo em tempo real que façam uso desta tecnologia.

Para se ter uma idéia do tempo de processamento de um codificador de vídeo usando uma técnica puramente fractal de compressão, suponha um vídeo de SDTV (*Standard Definition Television*) progressivo e monocromático com 480x720 *pixels* por frame, 8 bits/*pixel* e 24 *frames*/s sendo codificado à 4Mbits/s (que é o padrão para sinais SDTV digitais). Neste caso, cada frame codificado deveria ter, em média, 167 Kbits, o que significa aproximadamente 0,48 bits/*pixel*.

Mesmo ignorando a complexidade computacional de uma provável estimação de movimento, para esta taxa de compressão, o algoritmo QPIFS-Fisher implementado levou em torno de 2 horas para codificar cada *frame*. Cada segundo de filme levaria então em torno de dois dias de processamento (em uma plataforma semelhante à utilizada neste projeto).

Usando o algoritmo Híbrido proposto, este tempo cairia para aproximadamente 2 horas por segundo de filme (novamente sem contar a estimação de movimento), o que representa uma aceleração considerável, mas ainda longe do suficiente para tornar o procedimento utilizável para aplicações em tempo real.

No entanto, a maior qualidade de imagem e a crescente capacidade de processamento dos equipamentos estimula o aprofundamento do estudo do processo de codificação híbrida de vídeo fractal. Além disso, há ainda aplicações tais como as enciclopédias eletrônicas na qual a codificação é executada uma vez e a decodificação diversas vezes. Nesta classe de aplicações, os algoritmos fractais podem ser muito utilizados dada a sua característica de decodificação rápida.

11.3.1 - Codificador proposto

Seguindo esta linha, este trabalho apresenta uma extensão para vídeo progressivo do algoritmo híbrido de codificação de imagens estáticas apresentado na seção anterior.

Para realizar esta extensão, foi necessária a escolha de um algoritmo de compressão temporal. Como o esquema proposto utiliza a TWD como etapa inicial, a alternativa mais vantajosa encontrada foi a utilização do algoritmo de estimação/compensação de movimento no domínio da transformada wavelet encontrado na literatura recente e que apresenta bons resultados, que é uma das versões do MRME (*Multi-Resolution Motion Estimation*) propostas por Zan, Ahmad e Swamy [5]. O algoritmo MRME foi proposto originalmente por Zhang e Zafar [4] e é um dos mais utilizados nos artigos recentes sobre estimação de movimento para esquemas de compressão wavelet.

Na versão utilizada, a estimativa dos vetores de movimento das escalas mais baixas da wavelet é realizada através de um filtro de mediana na vizinhança do bloco de mesma posição espacial na escala anterior de forma a evitar a propagação de falsos vetores de movimento entre as escalas wavelet [5].

No esquema de vídeo proposto, após a etapa de compressão temporal, o resíduo de predição das Quadros P é codificado usando o SPIHT. Nestas imagens não foi utilizada a codificação fractal, visto que o erro de predição apresenta características parecidas com as subimagens de detalhes e,

portanto, não favoráveis à aplicação da codificação fractal. O diagrama em blocos do codificador proposto é apresentado na Fig. 11.2.

Com relação à codificação entrópica, foi utilizado um esquema simples de extração de redundância através da codificação diferencial dos vetores de movimento à partir do bloco anterior.

No entanto, não foi implementado nenhum outro sistema de codificação entrópica foi utilizado. Para estimar o ganho de compressão dessa etapa utilizou-se a estimativa da entropia de primeira ordem dos dados codificados.

Também quanto à estimação de movimento, foi se utilizada apenas a estimação de movimento com relação ao frame anterior (Quadros P). Isto porque, apesar do maior ganho de codificação apresentado pelas Quadros B, a complexidade computacional do codificador ficaria muito aumentada com sua utilização.

Com relação ao controle de taxa, utilizou-se um sistema de controle de taxa adaptativo que visa manter uma relação fixa entre a quantidade de bits gasta nas I- e nas Quadros P. Este controle realiza a adaptação do parâmetro de qualidade mínima e_c do codificador fractal de acordo com a taxa de bits almejada e a as taxa de bits usada pela parte fractal da Quadro I anterior.

Os resultados obtidos foram comparados ao MPEG-2 TM5 usando MP@ML com o algoritmo implementado pelo MPEG *Software Simulation Group* (MSSG) disponível em http://www.mpeg.org/MPEG/MSSG .

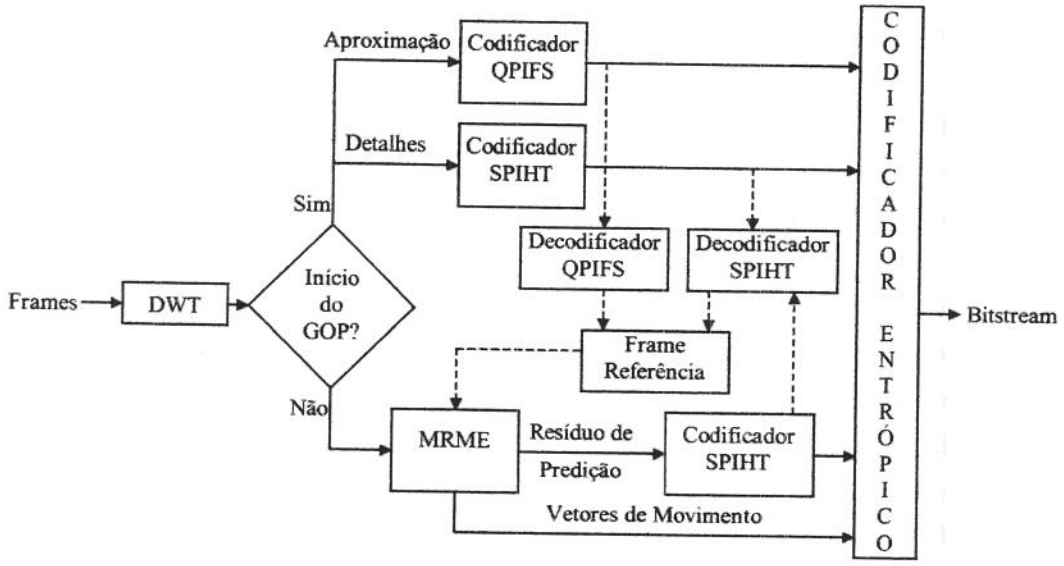


Fig. 11.2 – Diagrama de Blocos do Codificador de Vídeo Proposto

11.3.2 - Parâmetros de simulação

1) Híbrido:

- TWD: Biortogonal 9-7;
- Nível máximo de Quadtree: 5;
- Nível mínimo de Quadtree: 3;
- *Overlap*: 75%;
- Número de bits para S_i e O_i: 5 e 7 respectivamente;
- Taxa de bits de 2 Mbps;
- Taxa de frames: 30 fps;
- Tamanho de GOP: 4 frames;
- Estrutura de frames IPPP;
- Máximo VM: 10;

2) MPEG-2 TM5:

- Sem entrelace: vídeo progressivo;
- Taxa de bits: 2 Mbps;
- Tamanho de GOP: 15;
- Estrutura de frames: IBBP;
- Máximo VM: 10;
- Taxa de frames: 30 fps;

11.4 - Conclusões

Neste capítulo foram apresentados os algoritmos de codificação propostos para codificação de imagens e de vídeo. Também foram apresentados os parâmetros de simulação utilizados nos experimentos realizados. O próximo capítulo é dedicado a mostrar os resultados obtidos bem, como a sua análise.

Capítulo 12

Resultados Experimentais

12.1 - Introdução

Os resultados obtidos foram divididos novamente em duas partes, a primeira parte referente ao codificador de imagens que foi comparado com os sistemas de codificação SPIHT (baseado somente em TWD) e QPIFS (baseado somente em codificação fractal) para diversas imagens 512x512. Também foram realizadas algumas comparações com outros artigos publicados recentemente, como o codificador de Cardinal [17] e Li e Kuo [68]. A segunda parte dos resultados apresenta a comparação do algoritmo proposto com o algoritmo MPEG-2 MP@ML para diversas seqüências conhecidas da literatura científica de tamanho 720x480 *pixels* em escala de cinza, 30 *frames* por segundo, progressivo.

No caso de vídeo foi comparada somente a variação do PSNR no tempo para seqüências codificadas em 2Mbps. Não foram realizadas comparações quanto ao tempo de processamento, pois a implementação do algoritmo MPEG-2 utilizada nos testes está em linguagem C, enquanto que o codificador proposto está em Matlab script, inviabilizando uma comparação efetiva.

Nas Seções 12.2 e 12.3 são mostrados, respectivamente, os resultados subjetivos e objetivos do codificador de imagens proposto e dos esquemas base de comparação. Na Seção 12.3 TAMBÉM são realizadas outras comparações com resultados recentes da literatura para o codificador de imagens. E finalmente na Seção 12.4 são mostrados os resultados de vídeo.

12.2 - Codificador de Imagens: Resultados Subjetivos

O codificador híbrido proposto foi comparado à codificação fractal pura acelerada por Fisher (referenciada como "QPIFS") e à codificação wavelet pura (referenciada por "SPIHT"). OS três métodos são comparados em termos de tempo de codificação, qualidade subjetiva, PSNR e taxa de bits para diversas imagens adotadas pela literatura específica a várias taxas de bits.

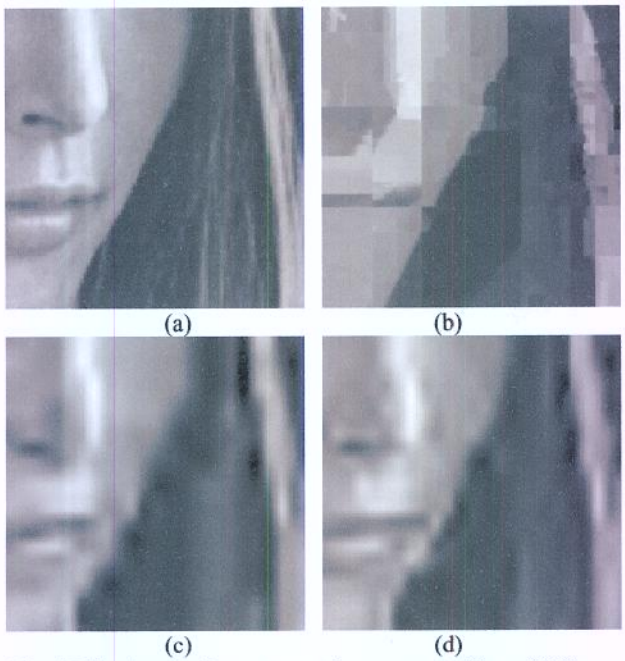
Os parâmetros de simulação adotados para os três casos são mostrados no capítulo anterior. A Fig. 12.1 mostra a imagem original *Lena* 512x512 a 8 bpp. Nas Figs. 12.2 a 12.4 são mostradas as imagens recuperadas pelos três métodos à taxa de 0,10 bpp, e na Fig. 12.5 a ampliação da área central dessas imagens.

Fig. 12.1 - Imagem *Lena* original, 512x512 *pixels* a 8 bpp.

Fig. 12.2 - Imagem *Lena* QPIFS, a 0,10 bpp – 25,32 dB.

Fig. 12.3 - Imagem *Lena* SPIHT, a 0,10 bpp – 25,44 dB.

Fig. 12.4 - Imagem *Lena* Híbrido, a 0,10 bpp – 27.61 dB.



(a) (b)

(c) (d)

Fig. 12.5 - Ampliações comparativas reconstruídas a 0,10bpp:
(a) Original; (b) QPIFS; (c) SPIHT; (d) Híbrido.

Observando as Figs. 12.2 a 12.4, devido à alta taxa de compressão, pode-se notar claramente as diferentes características dos métodos Fractal e Wavelet e como elas se refletem no método híbrido proposto. No esquema QPIFS, há uma grande ocorrência de efeito de bloco e algumas falhas de codificação de blocos, como os da parte de cima do chapéu e da boca de *Lena*. No entanto, há regiões onde a decodificação é muito superior à dos outros métodos, como os olhos e partes do cenário.

O SPIHT, por sua vez, causa um borramento e efeito de *ringing* (oscilações de brilho) em praticamente todas as regiões da imagem, ficando especialmente visível nas proximidades das bordas (descontinuidades) naturais da imagem. No entanto, em uma visão geral, a imagem recuperada se assemelha mais à original do que no caso QPIFS, onde as distorções são localizadas.

O Codificador Híbrido apresenta significativa melhora na qualidade subjetiva da imagem, pois evita completamente o efeito de bloco e reduz a perda de detalhes por borramento. O resultado também mostra uma redução no efeito *ringing*. Esta melhor qualidade é alcançada, pois o codificador proposto codifica melhor a sub-imagem passa-baixas atingindo uma melhor qualidade com uma taxa de compressão ainda maior que a do SPIHT, de forma que as camadas de detalhes também fiquem mais bem codificadas, o que leva uma melhoria de 2,2 dB em PSNR.

A ampliação mostrada na Fig. 12.5 torna mais evidentes todos os artefatos de compressão

citados: o borramento, o *ringing* e o efeito de bloco. Nesta figura pode-se notar que as distorções da imagem no caso do codificador híbrido são bem menores que nos outros casos, gerando uma imagem mais semelhante à imagem original.

Nas Figs. 12.6 a 12.9 serão mostradas as imagens recuperadas da codificação a 0,32 bpp da mesma imagem Lena. Nestas imagens pode-se observar que há uma menor intensidade dos artefatos de compressão apresentados na taxa anterior. No caso da codificação QPIFS não ocorrem mais erros de codificação de blocos, mas o efeito de bloco ainda é visível, especialmente em regiões planas como a face e os ombros de Lena.

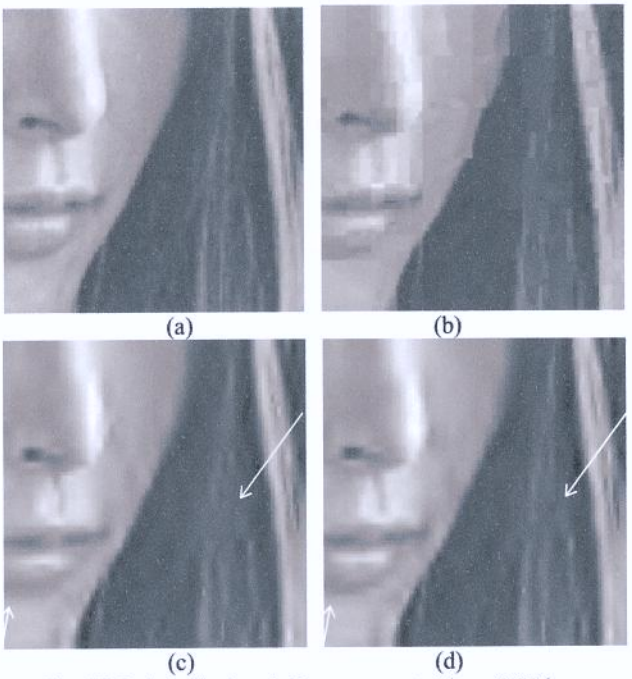
No caso da codificação SPIHT também pode-se notar uma significativa melhora na qualidade de imagem com relação ao caso anterior, alcançando uma excelente qualidade de imagem final. Os efeitos de borramento e *ringing* foram muito reduzidos somente sendo visíveis em alguns fios do penacho e na aba do chapéu de Lena.

Para o codificador híbrido o resultado foi análogo: houve uma grande redução dos artefatos de codificação gerando uma imagem de qualidade ligeiramente superior à do SPIHT. No entanto, dada a excelente qualidade de imagem reconstruída, esta diferença de qualidade só é visível na ampliação da Fig. 12.9.

Fig. 12.6. Imagem *Lena* PIFS, a 0,32 bpp – 31.19 dB.

Fig. 12.7. Imagem *Lena* SPIHT, a 0,32 bpp – 31,72 dB.

Fig. 12.8. Imagem *Lena* Híbrido, a 0,32 bpp – 32,10 dB.



(a) (b)

(c) (d)

Fig. 12.9. Ampliações de *Lena* reconstruídas a 0,32 bpp:
(a) Original; (b) QPIFS; (c) SPIHT; (d) Híbrido.

Na Fig. 12.9 foram inseridas algumas flechas indicando as regiões onde é possível observar as diferenças entre as imagens reconstruídas. Essas flechas indicam a ligeira distorção por borramento do lábio inferior e da textura do cabelo de Lena onde o codificador proposto obteve resultados melhores que o SPIHT.

Esse melhor desempenho do codificador híbrido é devido, novamente, ocorre uma melhor codificação da subbanda passa-baixas, fazendo com que sobrem mais bits para as camadas de detalhes da TWD. Neste ponto é interessante ressaltar que esse maior borramento por parte do SPIHT é praticamente constante por toda a imagem gerando, assim, um ganho de 0,4 dB no PSNR.

De forma semelhante à imagem Lena, também foram realizados experimentos com imagem *Goldhill* 512x512, mostrada na Fig.12.10. Nas Figs. 12.11 e 12.13 são mostradas as imagens recuperadas pelos três métodos testados na taxa de 0,10 bpp.

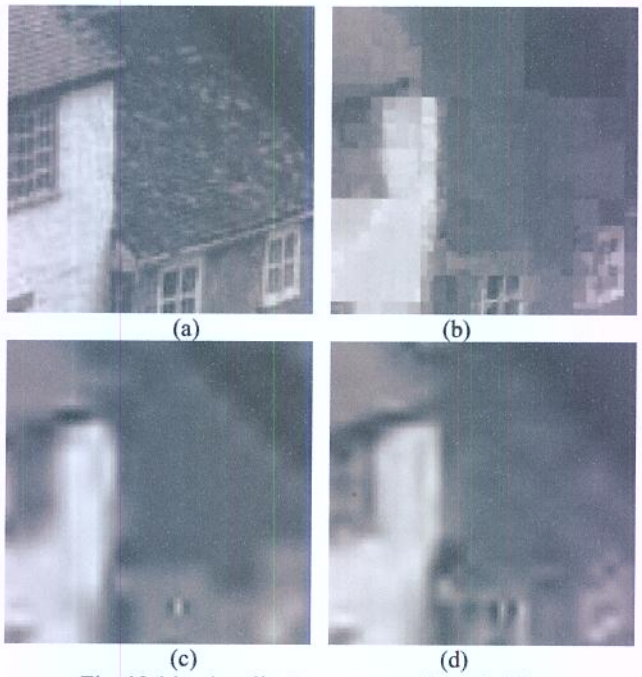
Analisando os resultados para a imagem *Goldhill* pode-se notar claramente que as características apresentadas anteriormente se mantêm: a codificação QPIFS apresenta falhas de codificação e efeito de bloco intenso, enquanto que as codificações SPIHT e Híbrida apresentam borramento e efeito *ringing*. Novamente, a qualidade geral da imagem recuperada pelo método híbrido é claramente superior ao outros métodos pela grande redução desses artefatos.

Fig. 12.10 – Imagem *Goldhill*. (a): Original 512x512 a 8 bpp.

Fig. 12.11 – Imagem *Goldhill* QPIFS a 0,10 bpp – 24,57 dB.

Fig. 12.12 – Imagem *Goldhill* SPIHT a 0,10 bpp – 24,76 dB.

Fig. 12.13 – Imagem *Goldhill* Híbrido a 0,10 bpp – 26,49 dB.



(a) (b)

(c) (d)

Fig. 12.14 – Ampliações reconstruídas a 0,10bpp
(a): Original; (b) QPIFS; (c) SPIHT; (d) Híbrido.

Os resultados para essa imagem são interessantes para a análise do comportamento dos codificadores, pois ela apresenta uma grande quantidade de bordas naturais (as divisões entre casas, entre paredes e telhados), regiões planas (homogêneas) e regiões com alto nível de detalhamento (telhados e janelas).

Para o codificador QPIFS (Fig. 12.11), pode-se notar que o melhor desempenho do codificador está na área das casas, com telhados e janelas. Isso ocorre pois a auto-similaridade nessas regiões é muito mais evidente fazendo com que o codificador encontre uma melhor casamento entre blocos-domínio e imagem. Já no caso da parte superior da imagem, a paisagem não apresenta auto-similaridade evidente, o que dificulta o trabalho do codificador. Com a restrição de taxa, essas regiões acabam ficando mais mal codificadas em relação à parte inferior da imagem.

O codificador SPIHT, por sua vez, apresenta um desempenho bom nas baixas freqüências do sinal. Dessa forma, a imagem recuperada se apresenta praticamente como uma versão borrada da imagem original. De uma forma geral, a imagem recuperada pelo SPIHT se assemelha mais à imagem original que a do QPIFS, pois os artefatos de compressão aqui são espalhados igualmente por toda a imagem enquanto que lá são concentradas nos blocos sem auto-similaridade evidente.

Por fim, a melhor qualidade de imagem apresentada pelo codificador proposto pode ser observada tanto na melhor distribuição das baixas freqüências (uma melhor definição das áreas homogêneas) como no aumento da quantidade de detalhes na imagem como um todo (que pode ser visto especialmente nas janelas e telhados). Assim, o codificador proposto apresenta uma boa combinação entre as vantagens de ambos os métodos, possibilitando uma melhor qualidade da imagem reconstruída e um ganho de 1,9 dB em PSNR.

As ampliações da região central das imagens recuperadas pelos três métodos, mostradas na Fig. 12.14, evidenciam essas características, de forma a torna mais clara as diferenças entre os resultados obtidos.

Nas Figs. 12.15 a 12.17 são mostrados os resultados para a taxa de 0,32 bpp, sendo que na Fig. 12.18 são mostradas as ampliações. Com a redução da taxa de compressão, a quantidade e a severidade dos artefatos se reduzem, resultando em imagens de melhor qualidade nos três casos. Ainda assim, no codificador QPIFS ainda é possível distinguir algum efeito de bloco, especialmente na parte superior direita da imagem.

Os codificadores SPIHT e Híbrido tem resultados muito semelhantes, sendo que o híbrido é ligeiramente melhor em alguns detalhes como a recomposição dos telhados. Aqui novamente o ganho entre os PSNRs do SPIHT e do Híbrido foi em torno de 0,4 dB.

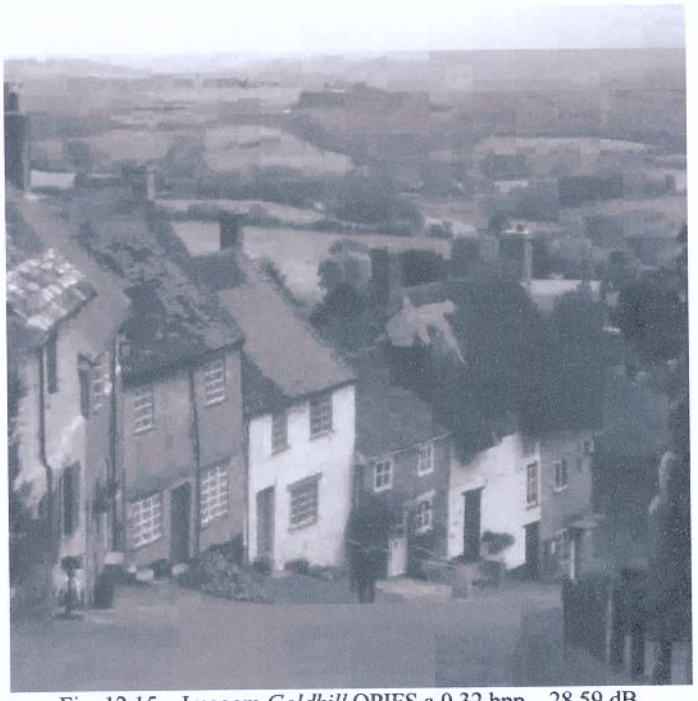


Fig. 12.15 – Imagem *Goldhill* QPIFS a 0,32 bpp – 28,59 dB.

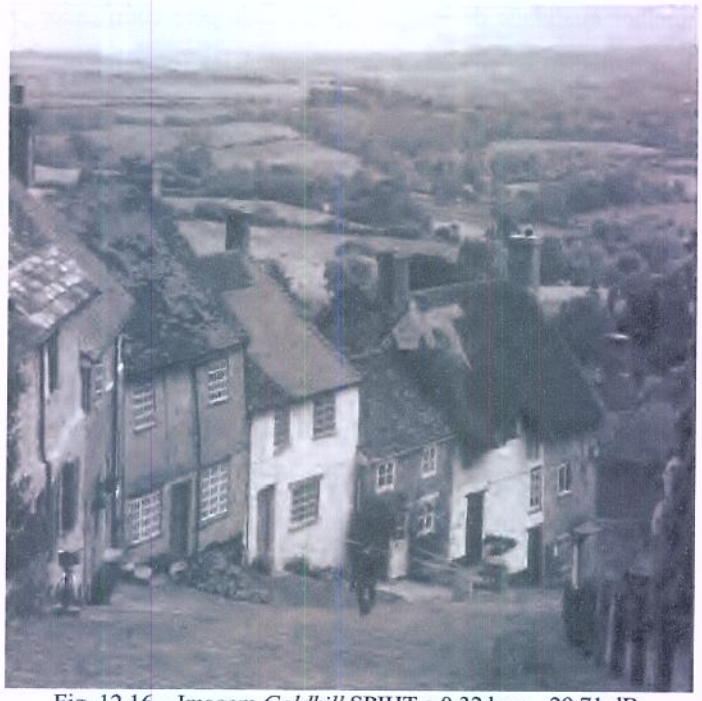


Fig. 12.16 – Imagem *Goldhill* SPIHT a 0,32 bpp – 29,71 dB.

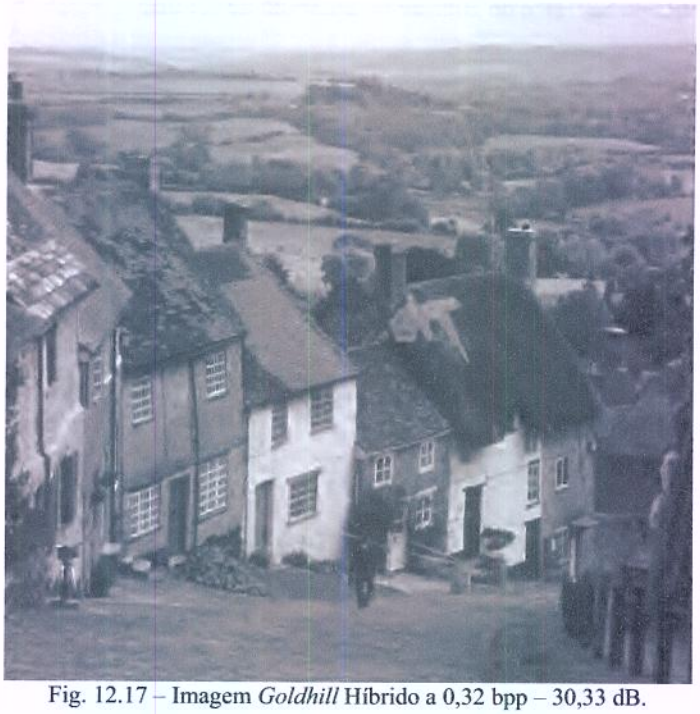


Fig. 12.17 – Imagem *Goldhill* Híbrido a 0,32 bpp – 30,33 dB.

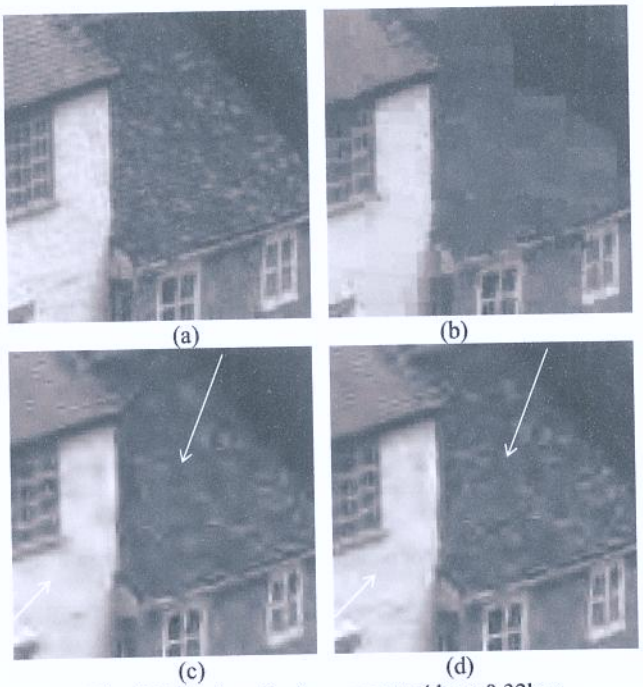


Fig. 12.18 – Ampliações reconstruídas a 0,32bpp
(a): Original; (b) QPIFS; (c) SPIHT; (d) Híbrido.

Utilizando as setas brancas ns ampliações mostradas na Fig. 12.18, procurou-se evidenciar mostradas algumas regiões onde o codificador Híbrido supera o SPIHT em termos de qualidade de imagem reconstruída. Como comentado anteriormente, essas melhorias são devido à melhor codificação da sub-imagem passa-baixas pelo codificador proposto.

Na próxima seção serão apresentados os resultados numéricos para todos os experimentos realizados com os três codificadores, incluindo a determinação do número de bits usados para quantizar os parâmetros s_i e o_i.

12.3 - Codificador de Imagens: Resultados Objetivos

Com o objetivo de ajustar os parâmetros de quantização fractal, várias simulações foram feitas para verificar como a quantização recomendada de [S_i, O_i] em [5,7] bits [7][17] opera em nosso codificador. As Figs 12.19 e 12.20 mostram as curvas de PSNR x [S_i, O_i] para *Lena* 512x512 a 0,1bpp. A Fig. 12.19 refere-se ao PSNR da imagem completa e a Fig. 12.20 refere-se ao PSNR entre as sub-imagens de Aproximação original e decodificada.

Como pode ser observado, todas as curvas de PSNR para a imagem reconstruída tem um comportamento côncavo, tendo um ponto de máximo em torno de [5,5] e [5,7]. Fora deste ponto, a distribuição de bits associados à Subbanda de Aproximação é menos eficaz, deixando bits demais ou de menos disponíveis para os detalhes piorando o desempenho geral do codificador.

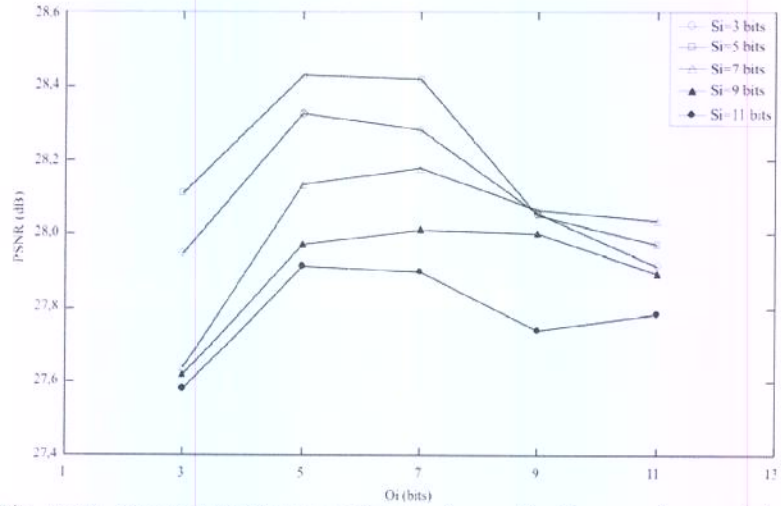


Fig. 12.19 - Curvas de PSNR por parâmetros de quantização para a imagem toda.

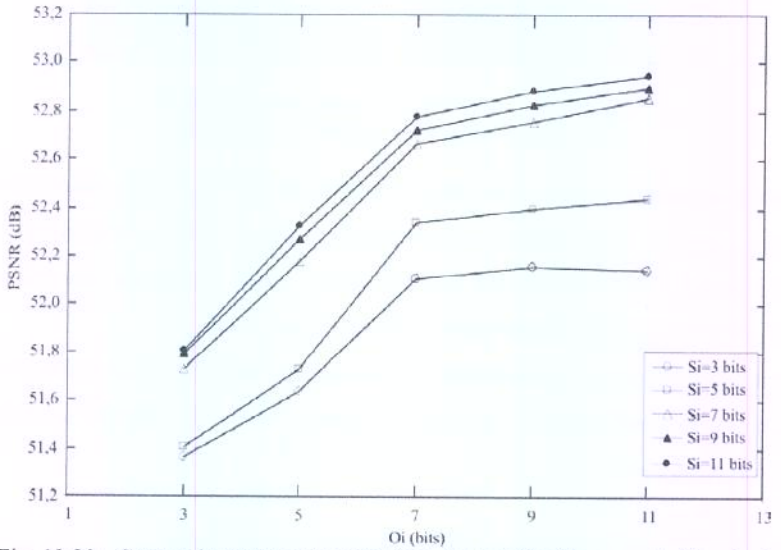


Fig. 12.20 - Curvas de PSNR por parâmetros de quantização para a Subbanda de Aproximação da Imagem *Lena* 512x512; 0,1bpp.

As tabelas 12.1 a 12.3 mostram o tempo de processamento e valores de PSNR para os três métodos a várias taxas de bits para três das 8 imagens testadas. O tempo de processamento indicado nas tabelas inclui os tempos de codificação e decodificação.

Os valores de PSNR do Codificador Híbrido são sempre melhores do que os do SPIHT em todos os casos. Comparado ao QPIFS, o método Híbrido é melhor à medida que a taxa de bits decai. A taxas de bits mais altas, QPIFS apresenta os melhores resultados entre os três procedimentos. A despeito de ser observado na literatura que a medida de PSNR nem sempre reflete as comparações subjetivas reais, nestes experimentos, os valores obtidos concordam com a análise subjetiva.

Tabela 12.1 – Resultados Numéricos para a Imagem *Lena*

Taxa de compressão	Taxa de bits(bpp)	Método	Tempo (s)	PSNR(dB)
80	0.10	QPIFS	00:12'34.5 "	25.32
		SPIHT	00:00'53.7 "	25.44
		Hybrid	00:02'06.8 "	27.61
60	0.13	QPIFS	00:13'08,6 "	27,79
		SPIHT	00:01'04.8 "	27.46
		Hybrid	00:02'14.3 "	29.06
40	0.20	QPIFS	00:25'03.2 "	29.54
		SPIHT	00:01'24.0 "	29.90
		Hybrid	00:02'41.5 "	30.63
25	0.32	QPIFS	01:06'00.7 "	31.19
		SPIHT	00:02'12.8 "	31.72
		Hybrid	00:03'27.9 "	32.10
12	0.66	QPIFS	02:35'16.8 "	35.73
		SPIHT	00:05'26.1 "	33.47
		Hybrid	00:06'39.0 "	33.52
10	0.84	QPIFS	03:20'34.6 "	36.58
		SPIHT	00:07'49.9 "	33.51
		Hybrid	00:08'34.2 "	33.90

Tabela 12.2 – Resultados Numéricos para a Imagem *Goldhill*

Taxa de compressão	Taxa de bits(bpp)	Método	Tempo (s)	PSNR(dB)
80	0.10	QPIFS	00:13'05.2 "	24.57
		SPIHT	00:00'57.2 "	24.76
		Hybrid	00:02'41.3 "	26.49
60	0.13	QPIFS	00:14'31.6 "	26.51
		SPIHT	00:01'05.9 "	26.12
		Hybrid	00:02'58.6 "	27.30
40	0.20	QPIFS	00:24'22.4 "	27.64
		SPIHT	00:01'36.7 "	28.02
		Hybrid	00:03'19.4 "	28.80
25	0.32	QPIFS	01:00'08.0 "	28.59
		SPIHT	00:02'14.1 "	29.71
		Hybrid	00:03'52.1 "	30.33
16	0.49	QPIFS	01:43'58.3 "	30.74
		SPIHT	00:03'37.2 "	31.38
		Hybrid	00:05'19.3 "	31.70
11	0.72	QPIFS	02:46'34.5 "	32.78
		SPIHT	00:06'14.0 "	32.42
		Hybrid	00:07'52.1 "	32.60

O desempenho do codificador proposto em termos de tempo de processamento está entre os esquemas SPIHT e QPIFS em todos os casos testados no que diz respeito ao tempo de codificação. O Codificador Híbrido apresenta uma média de 1,7 vezes o tempo de codificação do SPIHT, e o

QPIFS apresenta uma média de 17 vezes tempo de codificação do Híbrido. Portanto, o procedimento Híbrido reduz em média 94% do tempo de codificação, se comparado ao QPIFS.

O pior desempenho do método Híbrido, se comparado ao SPIHT surge da carga computacional da etapa de *matching* fractal e da partição *quadtree*. Por outro lado, o codificador Híbrido apresenta velocidade tão maior do que o QPIFS devido ao fractal *matching* e ao fato das *quadtrees* serem somente aplicadas à Subbanda de Aproximação.

Tabela 12.3 – Resultados Numéricos para a Imagem *Boat*

Taxa de compressão	Taxa de bits(bpp)	Método	Tempo (s)	PSNR(dB)
80	0.10	QPIFS	00:13'21.7 "	23.95
		SPIHT	00:00'57.5 "	23.72
		Hybrid	00:02'33.8 "	26.19
60	0.13	QPIFS	00:14'40.3 "	25,21
		SPIHT	00:01'04.1 "	25.42
		Hybrid	00:02'48.3 "	27.45
40	0.20	QPIFS	00:29'24.9 "	27.42
		SPIHT	00:01'29.5 "	27.80
		Hybrid	00:03'04.9 "	28.80
25	0.32	QPIFS	01:06'26.4 "	28.61
		SPIHT	00:02'21.2 "	29.76
		Hybrid	00:03'50.7 "	30.19
16	0.49	QPIFS	01:52'03.3 "	31.80
		SPIHT	00:03'00.6 "	31.48
		Hybrid	00:05'19.4 "	31.69
11	0.72	QPIFS	03:20'15.0 "	34.37
		SPIHT	00:06'05.9 "	32.23
		Hybrid	00:07'38.3 "	32.30

Comparações adicionais e considerações com relação a outros resultados publicados também foram realizadas. A Fig. 12.21 apresenta as curvas de taxa-distorção para os 3 codificadores testados e o codificador de Cardinal [17].

Cardinal [17] propôs um codificador fractal puro baseado num algoritmo heurístico que determina uma aproximação para a partição ótima do espaço de características. Os resultados experimentais de Cardinal foram tirados diretamente de [11, Fig. 4b] e também foram obtidos com as mesmas imagens 512x512 a 8bpp do *Greyset* padrão disponível no *website Waterloo Bragzone* [1]. As características R-D apontadas nas Tabelas 12.1 a 12.3 refletem-se na Fig. 12.21 e o codificador Híbrido proposto supera [17] por cerca de 3dB.

Essa grande diferença entre o codificador de Cardinal e os demais se deve principalmente, segundo nossas análises, à não utilização de transformações geométricas (isometrias) por parte de Cardinal. Essa não-utilização dificulta consideravelmente a procura pelas auto-similaridades,

fazendo com que a codificação fractal apresente um desempenho bem pior em termos de taxa-distorção.

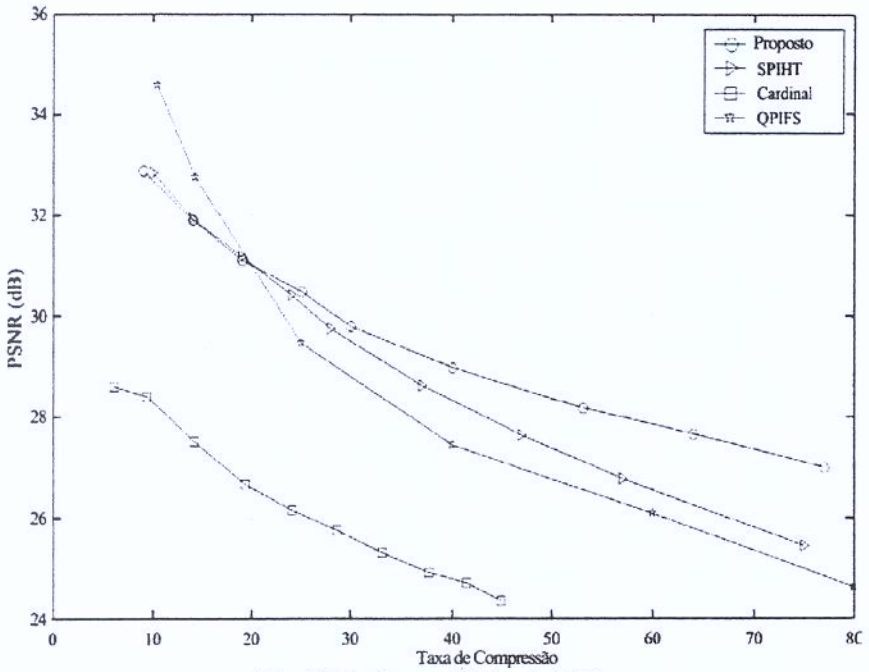


Fig. 12.21. Curvas Médias de R-D.

Também através da análise dessas curvas, pode-se notar que o desempenho do codificador proposto vai se afastando do SPIHT à medida que a taxa de compressão aumenta. Isso ocorre porque, como a sub-imagem passa-baixas tem um tamanho pequeno e, a baixas taxas de compressão, ambos os processos conseguem uma excelente aproximação para essa sub-imagem e a melhoria do incremento de bits passa a ser somente o acréscimo dos componentes de ata freqüência. A diferença entre os algoritmos é que, à medida que a taxa de compressão sobe, o codificador híbrido codifica melhor a passa-baixas, deixando mais bits para as altas freqüências.

Outra observação que pode ser feita é que o codificador QPIFS apresenta um desempenho melhor que os outros a baixas taxas de compressão. No entanto, a falta de otimização ainda faz com que o QPIFS apresente uma queda proporcionalmente maior, fazendo que o seu desempenho seja o pior à altas taxas de compressão. Esse resultado evidencia o grande potencial da compressão fractal ainda a ser explorado e a necessidade de maiores pesquisas para que esse potencial possa ser melhor explorado e levar, possivelmente, a padrões de compressão muito mais eficientes que os existentes hoje em dia.

Também foram realizadas algumas comparações teóricas entre o codificador proposto e resultados mais recentes da literatura. Em [68], Li e Kuo apresentaram um Codificador Híbrido Fractal-*Wavelet* eficiente, que usa o mapeamento contrativo fractal para predizer os coeficientes

wavelet e então codifica o resíduo de predição com um codificador wavelet *bitplane*. Os autores reportaram um ganho de 0,6-0,9 dB na imagem *Lena* 512x512, se comparado ao EZW (*Embedded Zerotree Wavelet Coder*) [42]. Nosso codificador Híbrido, por sua vez, supera o SPIHT por 0-2,4 dB para a mesma imagem. De acordo com Rao e Yip, o SPIHT é uma versão refinada e superior ao EZW [44].

Em [69], também é proposto um algoritmo eficiente para a redução do tempo de codificação, baseado numa predição de contraste zero que mantém a mesma qualidade de imagem do *Full Search*. Os resultados experimentais apresentados em [69] mostram que, aplicando a predição de contraste, o tempo de processamento do *Full Search* se torna duas vezes mais rápido; e o algoritmo DCT *Inner Product* [70] se torna 4 vezes mais rápido se comparado ao *Full Search*.

Nosso codificador Híbrido acelera a codificação fractal já acelerada (usando a classificação de domínio de Fisher) em aproximadamente 20 vezes. Como a classificação de Fisher usualmente reduz o tempo de processamento do *Full Search* em cerca de 20 vezes, pode-se estimar "grosseiramente" que o nosso codificador Híbrido seja aproximadamente 400 vezes mais rápido do que o *Full Search* com um decréscimo de cerca de 1-1,5 dB no PSNR.

12.4 - Resultados para Vídeo

Os resultados do codificador de vídeo proposto foram comparados com o algoritmo de compressão MPEG-2 para 5 diferentes seqüências de vídeo progressivo encontradas na literatura. Para cada seqüência foram codificados um total de 15 frames à taxa de 2 Mbits por segundo. As seqüências são de tamanho 720x480 *pixels*, 8bpp e 30fps. A Fig. 12.22 mostra o primeiro frame de cada seqüência testada:, *Flowers*, *Football*, *Kiel*, *Mobile* e *Susie*.

Estas seqüências foram escolhidas por apresentarem características bem diferentes entre si, possibilitando que o algoritmo seja testado sob condições variadas. A seqüência *Flower* é uma seqüência com muitos detalhes e com movimento relativamente lento e translacional. Nessa seqüência pode-se focalizar o teste na eficiência do algoritmo de compressão espacial. A seqüência Football pode ser dividida em duas regiões sendo a primeira o fundo (gramado) que é uma região praticamente homogênea e estática. A segunda região é composta pelos jogadores que se movem muito rapidamente e é o foco principal da cena. Nessa seqüência pode-se testar bem a eficiência da estimação de movimento sob diferentes condições.

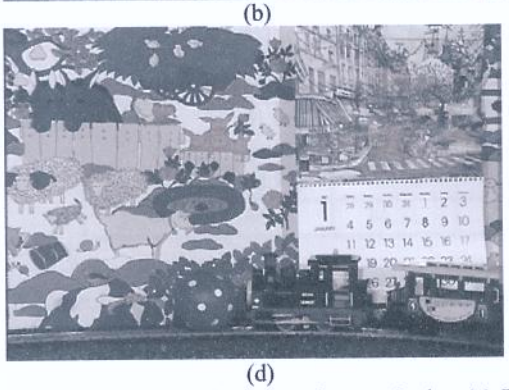
A seqüência Kiel além de apresentar detalhes em quase todas as regiões, também possui um movimento de aproximação da câmera (zoom) que é muito mal modelado pelos algoritmos de estimação/compensação tradicionais e, por isso, é usada para testar o desempenho desse algoritmo sob condições adversas. A seqüência Mobile contém vários detalhes que dificultam muito tanto a

compressão espacial quanto a Temporal. Com relação à compressão temporal, o movimento circular do trenzinho e a rotação da bola representam uma grande dificuldade. Já com relação à compressão temporal, os desenhos e letras podem ser vistos como descontinuidades nas imagens, o que reduz significativamente o desempenho do codificador. Por último, a seqüência Susie é composta por movimentos relativamente lentos e um fundo estático. Essas características concordam com o modelo assumido para a compressão de vídeo, o que leva à uma excelente qualidade dos quadros recuperados. Os resultados obtidos nas simulações são apresentados nas tabelas 12.4 e 12.5.

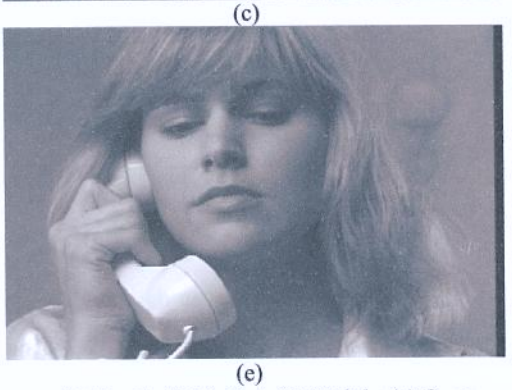
(a)

(b)

(c)



(d)



(e)

Fig. 12.22 – Primeiro Frame das seqüências: (a) *Flower*; (b) *Football*; (c) *Kiel*; (d) *Mobile*; (e) *Susie*.

Tabela 12.4 – Resumo dos Resultados numéricos PSNR x Frame – Parte 1

Frame no.	Seqüência *Flower*		Seqüência *Football*		Seqüência *Kiel*	
	MPEG2	Proposto	MPEG2	Proposto	MPEG2	Proposto
0	28,17	28,94	35,23	35,37	29,68	30,45
1	29,42	29,59	32,30	30,72	29,43	27,58
2	29,10	29,12	31,01	30,59	29,23	27,61
3	28,92	28,62	31,93	29,73	29,75	27,63
4	29,72	28,65	30,24	32,13	29,07	30,18
5	29,54	29,38	29,83	29,93	28,90	27,23
6	29,46	28,66	31,37	29,63	29,28	27,43
7	29,92	29,11	29,25	29,33	28,54	27,33
8	29,88	29,90	28,97	31,99	28,34	30,30
9	29,82	29,73	29,99	29,77	28,73	26,93
10	30,01	29,47	28,27	29,44	27,69	26,58
11	29,62	29,35	27,97	29,66	27,44	26,65
12	29,72	29,81	29,03	32,20	28,03	29,00
13	30,30	29,46	27,70	30,46	27,14	26,38
14	30,57	29,17	27,92	29,90	27,28	27,12
Média	29,61	29,26	30,07	30,72	28,57	27,89

Tabela 12.5 – Resumo dos Resultados numéricos PSNR x Frame – Parte 2

Frame no.	Seqüência *Mobile*		Seqüência *Susie*		Média	
	MPEG2	Proposto	MPEG2	Proposto	MPEG2	Proposto
0	27,36	27,89	38,84	38,75	31,85	32,28
1	27,43	24,40	38,16	38,56	31,35	30,17
2	26,91	24,67	38,14	38,24	30,88	30,05
3	27,28	24,64	37,74	38,49	31,12	29,82
4	26,49	27,66	38,55	37,90	30,81	31,30
5	26,26	24,24	38,50	38,57	30,60	29,87
6	26,37	24,53	38,40	39,40	30,98	29,93
7	25,96	24,30	38,97	39,01	30,53	29,82
8	25,62	27,04	38,85	38,99	30,34	31,64
9	25,79	24,44	38,74	38,67	30,62	29,91
10	25,28	24,40	38,63	38,29	29,98	29,64
11	25,09	24,36	38,63	38,30	29,75	29,67
12	25,33	27,32	38,41	38,59	30,10	31,39
13	25,43	23,99	37,86	37,56	29,69	29,57
14	25,63	24,43	37,71	37,33	29,82	29,59
Média	26,15	25,22	38,41	38,44	30,56	30,31

Como se pode observar das tabelas, o codificador proposto apresenta resultados médios muito próximos aos resultados do MPEG-2. Apesar disso, a variação do PSNR entre os *frames* das seqüências recuperadas pelo MPEG-2 é bem menor que a do codificador proposto.

Dois pontos são os principais motivos para que isso ocorra: o sistema de compressão temporal que ainda precisa de muitas melhorias para se aproximar do desempenho do codificador

MPEG-2, e o sistema de controle de taxa de bits que ainda necessita de um ajuste para tratar sinais que se distanciem muito do modelo assumido para as imagens.

Apesar de ainda haver uma grande quantidade de parâmetros e algoritmos para otimizar, os resultados para o codificador proposto podem ser considerados muito promissores, tendo em vista a semelhança com os resultados do MPEG-2 TM5.

As Fig. 12.23 a 12.28 mostram os gráficos de evolução do PSNR das seqüências testadas para ambos os sistemas: Posposto e MPEG-2 TM5.

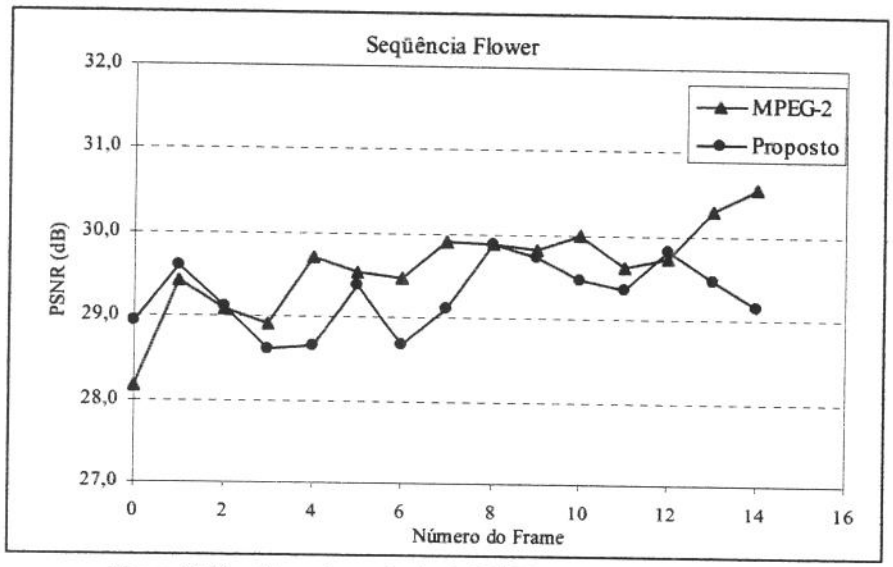


Figura 12.23 – Curva de evolução do PSNR para a seqüência *Flower*.

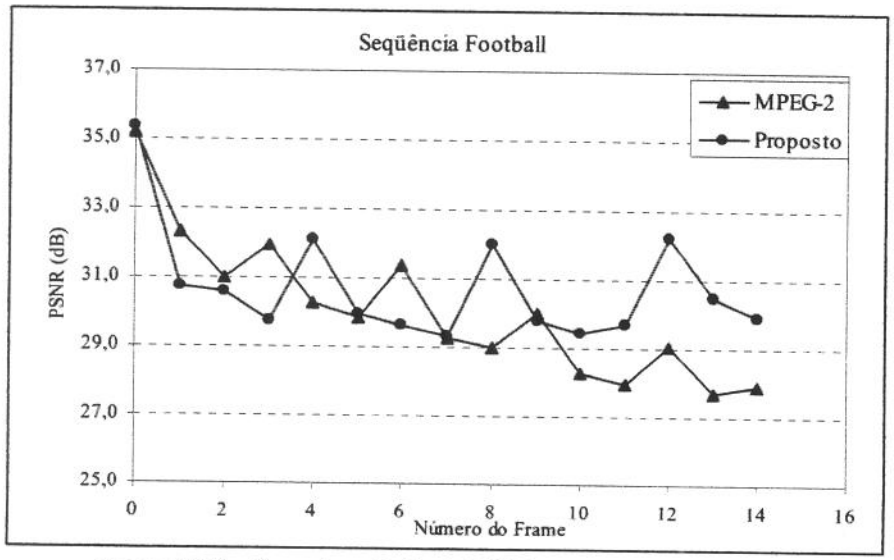


Figura 12.24 – Curva de evolução do PSNR para a seqüência *Football*.

Através da observação da evolução do PSNR para a seqüência Flower, mostrada na Fig. 12.23, pode-se notar que o sistema proposto apresenta um desempenho muito próximo ao do MPEG-2 para a maior parte da seqüência. Analisando esse comportamento levando em conta a característica de movimento lento e translacional da seqüência, podemos concluir que o sistema de compressão espacial de ambos os sistemas estão apresentando um desempenho similar.

Já na Fig. 12.24 é apresentado o gráfico para a seqüência Football. Nesse gráfico já se pode ver que o sistema proposto apresenta algumas variações periódicas no PSNR. Esses picos ocorrem primariamente nas Quadros I (lembrando que a estrutura do GOP para o codificador proposto é IPPP). Isso ocorre devido à uma falha no controle de taxa de bits, que não está conseguindo ainda equalizar a qualidade entre os quadros da seqüência.

Apesar disso, o desempenho do codificador proposto é ligeiramente superior ao do MPEG-2. Analisando as características da seqüência, pode-se concluir que isso ocorre pois a seqüência apresenta muitas partes de baixo movimento e grande quantidade de detalhes (o gramado) que reduz o desempenho da transformada DCT usada no padrão MPEG-2.

Na Fig. 12.25 é mostrado o gráfico para a seqüência Susie, no qual pode-se ver em que condições o codificador proposto praticamente se iguala ao MPEG-2 em termos de desempenho. Isto ocorre porque essa imagem apresenta movimentos muito lentos e localizados, sendo que grande parte da imagem é homogênea e está completamente parada. Com isso, ambos os algoritmos conseguem realizar uma boa compressão temporal, fazendo com que os macroblocos-erro resultantes da compensação de movimento sejam praticamente nulos, reduzindo assim a distorção pela quantização.

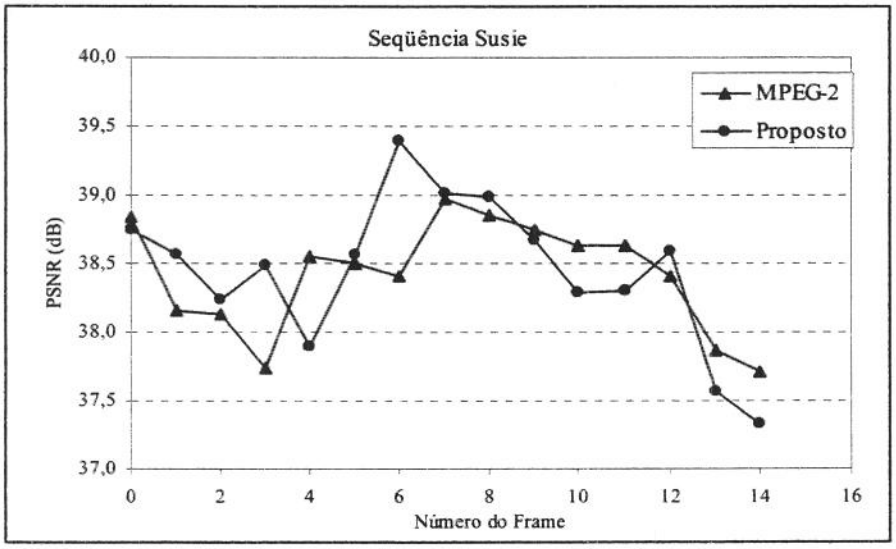


Figura 12.25 – Curva de evolução do PSNR para a seqüência *Susie*.

Nas Figs. 12.26 e 12.27 são mostradas as evoluções para as seqüências *Mobile* e *Kiel*.

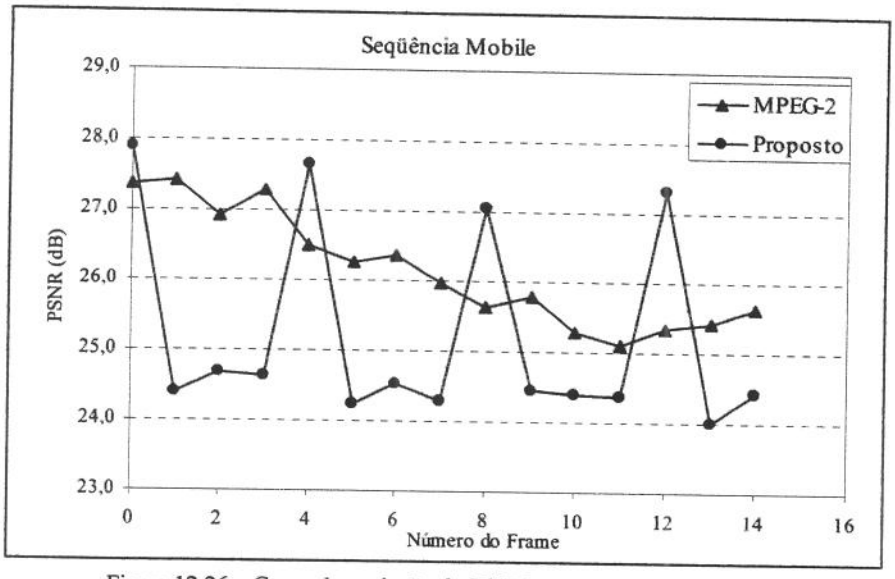


Figura 12.26 – Curva de evolução do PSNR para a seqüência *Mobile*.

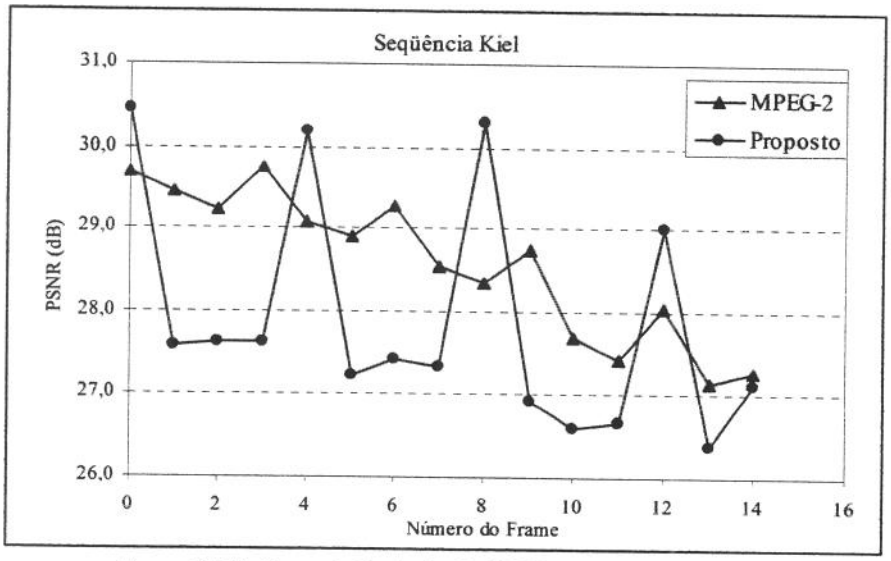


Figura 12.27 – Curva de Evolução do PSNR para a seqüência *Kiel*.

Nessas duas seqüências ocorre um agravamento do efeito notado na seqüência Football, apesar de a média se manter similar ao MPEG-2, o codificador proposto mostra um comportamento muito irregular, com grandes picos de PSNR nos Quadros I e grandes quedas nos Quadros P. A causa desse efeito pode ser encontrada comparando-se as características das duas seqüências. Ambas as seqüências apresentam um tipo de movimento que reduz significativamente o

desempenho da compressão temporal. Com isso os macroblocos-erro que entram na compressão temporal nos Quadros P apresentam muito mais energia do que eles teriam normalmente, exigindo assim que o controle de taxa de bits se adapte à essa nova característica. Nesse ponto entra uma das grandes diferenças entre os dois algoritmos, pois o MPEG-2 possui um controle de taxa bem estável e otimizado o que faz com que ele se adapte rapidamente à essa situação enquanto que o codificador proposto não consegue se adaptar causando assim a distorção.

Além do controle de taxa de bits, essa característica também poderia ser reduzida através da melhoria do algoritmo de estimação/compensação de movimento de forma a reduzir a energia do macrobloco-erro resultante da compressão temporal e, portanto, facilitando a tarefa da compressão espacial. A implementação destes dois pontos pode elevar a qualidade dos *frames* do algoritmo proposto acima da qualidade do MPEG-2 e ainda apresentando outra vantagem inerente ao uso da TWD: a ausência do efeito de bloco característico do MPEG-2 em altas taxas de compressão.

12.5 - Conclusões

A codificação fractal de imagens tem aplicações limitadas devido ao exaustivo tempo de codificação inerente. Com o objetivo de resolver esta questão mantendo a qualidade visual, esta pesquisa apresentou um novo codificador Híbrido rápido de imagens estáticas que combina a velocidade da TWD à qualidade de detalhes da imagem da compressão fractal.

O método proposto foi comparado à técnica de codificação fractal pura (QPIFS) e à codificação wavelet pura (SPIHT). Uma melhora significativa na qualidade subjetiva foi alcançada, evitando efeitos de *blurring* e não gerando quaisquer artefatos de blocagem a várias taxas de bits e para várias imagens; fato corroborado pelas medidas objetivas de PSNR obtidas.

A respeito do tempo de codificação, o desempenho do método proposto ficou entre o SPIHT e o QPIFS, apresentando uma média de 94% de redução do tempo de codificação, se comparado ao segundo. Desta forma, conclui-se que o método Híbrido proposto eficientemente reduz o exaustivo tempo de codificação fractal, apresentando também melhor desempenho visual. Esta última vantagem também foi obtida com relação ao SPIHT.

Experimentos e considerações adicionais também foram feitos comparando o codificador proposto com os algoritmos de Cardinal [17], Li e Kuo [68] e Lai *et. al.*[69]. Além disso, a característica da transmissão progressiva da wavelet é levemente reduzida, porém mantida.

Métodos fractais são convenientes para aplicações de arquivamento, tais como enciclopédias digitais, onde uma imagem é codificada uma vez e decodificada muitas vezes. Por outro lado, técnicas wavelet são convenientes para aplicações que requerem codificação rápida, tais

como a comunicação via Internet [10]. Métodos Híbridos, como o proposto neste trabalho, permitiriam a extensão de técnicas fractais para comunicações rápidas.

Com relação ao codificador de vídeo, apesar de ainda estar em estágio inicial de desenvolvimento, o algoritmo implementado apresenta resultados muito promissores, visto que o PSNR médio para as seqüências apresentam valores muito próximos ao do algoritmo MPEG-2 TM5, que é um algoritmo altamente otimizado e com excelentes resultados.

Analisando os resultados obtidos chegou-se a conclusão que as etapas de controle de taxa de bits e estimação/compensação de movimento são as maiores candidatas a essa otimização, visto que o desempenho do codificador proposto foi mais irregular nas seqüências em que essas etapas foram mais exigidas.

Capítulo 13

Conclusões e Sugestões para Trabalhos Futuros

13.1 - Introdução e Objetivos

A importância e a participação da imagem digital vêm sofrendo um crescimento exponencial nos últimos anos. Os adventos tecnológicos como os *scanners*, impressoras gráficas, *webcams* e câmeras digitais introduziram a imagem digital de modo irreversível na vida cotidiana.

Essa universalização da tecnologia digital leva a crescentes investimentos em pesquisas pela evolução dos algoritmos de codificação de imagens (e vídeo) e pelo desenvolvimento de novas técnicas que possam apresentar novas soluções para o problema da codificação. Dentro desse contexto, as tecnologias fractal e *wavelet* têm recebido cada vez mais atenção por parte dos pesquisadores da área de codificação de imagens e vídeo digitais.

A TWD consiste na aplicação de filtragens sucessivas (e inversíveis) gerando uma análise de tempo-freqüência do sinal original. Os sinais gerados por essa análise apresentam uma grande concentração de energia nas baixas freqüências e uma estrutura espacial peculiar que facilitam a representação compacta do sinal original. As principais características das transformadas *wavelet* são a transmissão hierárquica, algoritmos rápidos de decomposição e síntese, um boa relação de taxa-distorção, e a não-introdução artefatos de bloco. Devido a essas vantajosas características, a possibilidade da utilização da TWD foi incluída em importantes padrões recentes de compressão de imagem e vídeo, tais como o JPEG2000, MPEG-4 e MPEG-7.

Já a compressão fractal se constitui em representar os blocos da imagem original através da similaridade com outros blocos de tamanho maior dentro da própria imagem. A auto-similaridade é um conceito herdado dos processos de geração de fractais que representa a semelhança da imagem com si mesma em diferentes escalas. A codificação fractal apresenta como principais vantagens a rápida decodificação, arquitetura altamente paralelizável, curvas de taxa-distorção competitivas e, em especial, o fato de representar uma nova perspectiva sobre o problema da codificação digital de imagens. Por ser uma técnica muito recente e apresentar um alto potencial em relação à compressão de imagens, se constitui como um vasto campo de pesquisa.

Além de representarem novas linhas de pesquisa, essas técnicas possuem alguns conceitos em comum, como a exploração de semelhanças entre escalas diferentes da imagem, o que facilita o

desenvolvimento de técnicas híbridas, que procurem combinar as suas vantagens na busca de algoritmos mais eficientes.

Dentro desse contexto, o objetivo deste trabalho foi realizar um estudo profundo sobre essas duas técnicas recentes de codificação de forma a promover o desenvolvimento de algoritmos de codificação que combinem a velocidade e a simplicidade das técnicas wavelet com a qualidade da imagem fractal, obtendo, assim, um sistema de codificação de imagens com relação de taxa-distorção melhorada e boa velocidade de codificação que permitisse o desenvolvimento de uma extensão para a codificação de vídeo.

Seguindo essa linha, foram apresentados dois algoritmos, o primeiro para codificação de imagens que obteve um valor de PSNR de 0 a 2,4dB acima do algoritmo SPIHT e em uma redução média de 94% no tempo de processamento em relação ao QPIFS. O segundo algoritmo desenvolvido foi um algoritmo de codificação de vídeo que, apesar de se encontrar ainda em fase inicial, apresentou um desempenho médio similar ao do MPEG-2 TM5 para seqüências progressivas.

13.2 - Resumo e Organização da Tese

A apresentação dessa tese foi organizada em quatro partes principais, sendo que as três primeiras compõem o embasamento teórico resumido de cada uma das técnicas envolvidas e a última parte apresenta os resultados, análises e conclusões.

Na primeira parte foram apresentadas as bases matemáticas dos algoritmos fractais, iniciando, no capítulo 2, com os conceitos básicos de fractais, seu princípio de geração e suas bases matemáticas, passando pela revisão dos conceitos de topologia matemática utilizados.

No terceiro capítulo foram apresentados o conceito de auto-similaridade em imagens naturais e o esquema de codificação de imagem desenvolvido por Jacquin [3], o PIFS – *Partitioned Iterated Function Systems* – que é o sistema base para praticamente todos os esquemas fractais encontrados na literatura.

Finalizando a primeira parte da tese, o Capítulo 4 foi dedicado à apresentação de algoritmos recentes de codificação fractal acelerado pela técnica de classificação de domínios. Nessa técnica propõe-se reduzir a quantidade de comparações necessárias na etapa de procura por auto-similaridades o que acelera o processo de codificação.

A segunda parte foi dedicada à transformada wavelet e suas características. Seguindo a evolução da TW, foi apresentada no capítulo 5 as transformadas wavelet contínuas, passando pela série wavelet e chegando à TW discreta e ao algoritmo de Mallat [28] para o cálculo da TWD.

O sexto capítulo aborda a análise multi-resolução para os casos ortogonal uni e bi-dimensional, que consistem na base matemática da decomposição e da síntese através da TWD, bem como as definições dos conceitos de sub-bandas de aproximação e detalhamento comumente utilizadas em wavelet. Também se analisa o efeito da ortogonalidade e suas restrições discutindo-se a necessidade do relaxamento dessas restrições para se atingir *wavelets* biortogonais, apresentadas no Capítulo 7, que apresentam características de suporte compacto, simetria e bom compromisso entre regularidade e comprimento do filtro sendo, assim, mais propícias para a aplicação em imagens.

No Capítulo 8 completou-se a fundamentação matemática da transformada wavelet (parte 2) com a discussão sobre as características das *wavelets* que são mais interessantes no âmbito da compressão de imagens, como a regularidade e a simetria. Também foram apresentadas nesse capítulo alguns métodos de seleção de *wavelets* e o algoritmo de codificação SPIHT – *Set Partitioned in Hierarchical Trees* – que foi utilizado como base de comparação para os resultados obtidos.

A terceira parte, composta pelos capítulos 9 e 10, introduziu os conceitos básicos da codificação de vídeo e a terminologia utilizada na presente pesquisa (Capítulo 9) culminando com a descrição do algoritmo de estimação de movimento multi-resolução de Swammy *et. al.* [5] (Capítulo 10), que é uma evolução do algoritmo original desenvolvido por Zang e Zafar [4], e que foram incorporadas no algoritmo de codificação de vídeo desenvolvido durante a pesquisa.

A descrição mais detalhada dos algoritmos propostos para codificação de imagem e para codificação de vídeo e seus resultados foram o tema da parte final da tese. Os algoritmos desenvolvidos e os fundamentos que levaram à sua construção foram apresentados no Capítulo 11. No Capítulo 12 foram apresentados os resultados experimentais e a análise comparativa em termos de qualidade visual, PSNR, taxa de compressão e tempo de processamento para todos os algoritmos de codificação. Também são apresentados e analisados os resultados do codificador de vídeo em comparação com o MPEG-2 TM5.

No presente capítulo apresenta-se, à luz dos resultados obtidos e de sua análise, as conclusões que puderam ser extraídas e as perspectivas de melhorias e desenvolvimentos futuros que podem colaborar para a ampliação e refinamento desse trabalho.

13.3 - Comentários e Conclusões

Os resultados obtidos foram divididos em duas partes, a primeira parte referente ao codificador de imagens que foi comparado com os sistemas de codificação SPIHT (baseado em

somente em TWD) e QPIFS (baseado em somente em codificação fractal) para todas as imagens de tamanho *512x512 pixels* do conjunto *Waterloo Bragzone Greyset 2*, disponível na internet no site: http://links.uwaterloo.ca. Também foram realizadas algumas comparações com outros artigos publicados recentemente, como o codificador de Cardinal [17] e Li e Kuo [68]. A segunda parte dos resultados apresentam a comparação do algoritmo proposto com o padrão MPEG-2 MP@ML para diversas seqüências conhecidas da literatura científica de tamanho 720x480 *pixels* em escala de cinza, 30 *frames* por segundo, progressivo.

No caso de vídeo foi comparada somente a variação do PSNR no tempo para seqüências codificadas em 2Mbps. Não foram realizadas comparações quanto ao tempo de processamento, pois a implementação do algoritmo MPEG-2 utilizada nos testes está em linguagem C que está enquanto que o codificador proposto está em Matlab script inviabilizando uma comparação efetiva.

Analisando os resultados obtidos para o codificador de imagens, o algoritmo proposto apresentou uma melhoria significativa de qualidade com relação aos algoritmos puros, especialmente a taxas de compressão mais altas. Nesses casos, o algoritmo proposto alcançou qualidade visual superior combinando a boa recuperação de detalhes do algoritmo fractal com a melhor qualidade geral do algoritmo SPIHT, como visto nas Figs. 12.2 a 12.18.

Quanto à qualidade objetiva (PSNR) o esquema proposto também conseguiu melhorias no PSNR de até 2,4 dB sobre o SPIHT e até 2,2 dB sobre o QPIFS, sendo que o algoritmo proposto só foi suplantado pelo QPIFS na taxas de compressão muito baixas (maiores taxas de bits).

No aspecto de velocidade computacional, também o algoritmo proposto apresentou resultados muito bons, uma vez que obteve uma redução média de 94% do tempo de processamento do algoritmo fractal se aproximando muito do algoritmo SPIHT. Como o algoritmo SPIHT é implementável em tempo real, esta aproximação se constitui em passo de grande importância no caminho de desenvolver aplicações em tempo real para codificadores que utilizem a tecnologia fractal.

Também foram realizadas outras comparações teóricas com resultados publicados em artigos recentes da área, como o algoritmo de Cardinal [17] e de Li e Kuo [68]. Estas comparações foram efetuadas baseando-se diretamente nos resultados publicados, sem que fossem implementados os algoritmos. Ainda assim, em ambos os casos o algoritmo proposto também apresentou um desempenho significativo, superando o algoritmo fractal puro de Cardinal em mais de 3 dB em PSNR e apresentando um claro ganho de qualidade com relação ao codificador híbrido fractal-*wavelet* de Li e Kuo, comprovando assim a qualidade do sistema de codificação de imagens proposto.

Também foram apresentados os resultados iniciais do algoritmo de codificação de vídeo proposto. Esses resultados foram comparados aos resultados do algoritmo MPEG-2 TM5, que é o padrão de uma gama muito ampla de aplicações comerciais atuais, como DVDs, câmeras digitais, televisão digital e HDTV.

Nestes experimentos, o algoritmo proposto apresentou resultados muito bons, praticamente igualando a qualidade média obtida pelo MPEG-2 em seqüências progressivas de SDTV, apesar de ainda necessitar de uma maior estabilidade, especialmente no tocante ao controle de taxa e compressão temporal, que ainda apresenta muitas oscilações de desempenho de acordo com a seqüência a ser codificada.

No entanto, esses resultados podem ser considerados muito promissores, uma vez que o codificador de vídeo proposto ainda se encontra em fases iniciais de desenvolvimento podendo obter resultados muito mais significativos através da incorporação de algoritmos mais avançados para as etapas de compressão temporal e codificação entrópica.

13.4 - Sugestões para Trabalhos Futuros

Através do estudo dos princípios de funcionamento, dos resultados obtidos e das análises realizadas, pode-se indicar alguns possíveis melhoramentos futuros com base nesse trabalho:

- Estudo da variação do desempenho dos algoritmos com a variação das características de seleção da transformada wavelet discreta usada para a decomposição das imagens;
- Aplicação de técnicas de *lifting* na decomposição e síntese *wavelet* de forma a acelerar o processo de filtragem;
- Estudo e aplicação de outros algoritmos de classificação fractal na sub-banda de aproximação de forma a adquirir desempenho melhorado em relação à qualidade de codificação e/ou tempo de processamento;
- Desenvolvimento e incorporação de algoritmos de codificação entrópica dedicados que possam apresentar uma maior remoção da redundância dos dados, propiciando uma maior taxa de compressão para a mesma qualidade;
- Incorporação de técnicas avançadas de estimação/compensação de movimento de forma a tornar o algoritmo de codificação de vídeo mais eficiente;
- Estudo e aplicação de algoritmos de controle de taxa de codificação de vídeo;

Assim, foram apresentados os resultados e as conclusões do presente trabalho, assim como as sugestões de futuros trabalhos que possam se encaixar dentro da pesquisa desenvolvida durante essa tese de doutorado.

Espera-se que, de alguma forma, esse trabalho tenha colaborado para expor o alto potencial tanto dos sistemas de compressão fractal puro como em conjunto com as transformadas wavelet na aplicação de codificação de sinais, imagem e vídeo digitais e que ele possa ajudar a despertar o interesse de mias pesquisadores para esta área que apresenta tão boas perspectivas de resultados futuros.

Bibliografia

[1]– J. E. Hutchinson – "Fractal and Self-Similarity" – *Indiana University Mathematics Journal*, Vol. 35, No. 5, 1981.

[2]– M. F. Barnsley, J. H. Elton and D. P. Hardin – "Recurrent Iterated Function Systems" – *Constructive Approximation*, Vol. 5, No. 1, pp 3 – 48, 1989.

[3]– A. Jacquin – "A Fractal Theory of Iterated Markov Operators with Applications to Digital Image Compression" – PhD. Thesis, Georgia Institute of Technology, August, 1989.

[4]– Y. -Q. Zhang and S. Zafar – "Motion-Compensated Wavelet Transform Coding for Color Video Compression" – IEEE Transactions on CSVT, vol. 2, no. 3, pp. 285-296, Setembro, 1992.

[5]– J. Zan, O. Ahmad and M. N. S. Swamy – "New techniques for Multiresolution Motion Estimation" – *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 9, pp. 793-802, Setembro, 2002.

[6]– N. Lu – "Fractal Imaging" – Ed. Academic Press – 1997;

[7]– Yuval Fisher – "Fractal Image Compression, Theory and Application" – Ed. Springer-Verlag New York, Inc. – 1995;

[8]– M. F. Barnsley– "Fractals Everywhere 2nd edition" – Ed. Morgan Kaufmann – Academic Press, San Diego, 1993

[9]– E. L. Lima – "Espaços métricos" - Ed. Projeto Euclides – Rio de Janeiro, 1993.

[10]– S. T. Welstead – "Fractal and Wavelet Image Compression Techniques" – Ed. SPIE Optical Engineering Press, Washington, 1999.

[11]– Hygino H. Domingues – "Espaços métricos e introdução à topologia" - Ed. Atual Ltda – São Paulo, 1982.

[12]– S. Lipschutz – "Teoria dos conjuntos" – Ed. McGraw-Hill do Brasil – tradutor: Fernando Vilain Heusi da Silva – São Paulo, 1972.

[13]– B. Mandelbrot – "The Fractal Geometry of Nature" – Ed. W. H. Freeman and Company, New York, USA, 1983.

[14]– P. R. Massopust – "Fractal Functions, Fractal Surfaces, and Wavelets" – Academic Press, San Diego, 1995.

[15]– W. Rudin – "Real and Complex Analysis" – Ed. Maximillian, New York, 1972.

[16]– M. Polveri and M. Nappi – "Speed-Up in Fractal Image Coding: Comparison of Methods" –*IEEE Transactions on Image Processing*, Vol. 9, No. 6, pp. 1002-1009, June 2000.

[17]– J. Cardinal – "Fast Fractal Compression of Greyscale Images" – IEEE Transactions on Image Processing, vol. 10, no. 01, pp. 159-164, Jan., 2001.

[18]– E. W. Jacobs, R. D. Ross and Y. Fisher – "Fractal-based Image Compression II" – *Technical Repot 1362*, Naval Ocean Systems Center, San Diego, 1990.

[19]– E. W. Jacobs, R. D. Ross and Y. Fisher – "Fractal Image Compression using Iterated Transforms" – *Technical Report 1408*, Naval Ocean Systems Center, San Diego, 1991.

[20]– D. Saupe – "Fractal Image Compression by Multi-Dimensional Nearest Neighbor Search" – *Proceedings DCC'95 Data Compression Conference*, March 1995.

[21]– R. F. Sproull – "Refinements to Nearest Neighbor Searching in K-Dimensional Trees" – Algorithmica, vol. 6, pp. 579-589, 1991.

[22]– K. R. Castleman, "Digital Image Processing", Ed. Prentice Hall, New Jersey, USA 1996.

[23]– A. Grossman and J. Morlet, "Decompsition of Hardy Functions into square integrable wavelets of constant shape", in *SIAM J. Applied Mathematics*, vol. 15, pp.726-736, 1984.

[24]– C. F. Chui, "An introduction to wavelets", Academic Press Inc, San Diego, USA, 1992.

[25]– I. Daubechies, "Ten lectures on wavelets", SIAM, Philadelphia, USA, 1992.

[26]– M. C. Q. Farias and A. Lopes, "Aplicação da transformada wavelet na compressão de imagens", Dissertação de Mestrado, FEEC/UNICAMP, Junho 1998.

[27]– J. W. Woods and S. D. O'Neill, "Subband coding of images", in *IEEE Transaction ASSP*, vol.34, pp.1278-1288, 1986.

[28]– S. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation", in *IEEE Transactions on PAMI*, vol. 11, pp. 674-693, 1989.

[29]– A. L.M. C. S. Silva – "Procedimentos para Método Híbrido de Compressão de Imagens Digitais Utilizando Transformadas Wavelet e codificação Fractal", Tese de Doutorado, Faculdade de Engenharia Elétrica e Computação, Unicamp, Maio, 2005.

[30]– E. H. Adelson, E. Simincelli and R. Hingorani, "Orthogonal pyramid transforms for image coding", in Proceedings of SPIE, vol.845, pp.50-58, Out ubro, 1987.

[31]– B. Porat, "Digital processing of random signals", Ed. Prentice Hall, Pennsylvania, 1992.

[32]– E. H. Adelson; P. Burt – "The Laplacian Pyramid as a Compact Image Code" – IEEE Transactions Commun, vol. 31, pp. 482-540, Outubro, 1983.

[33]– M. Antonini, M. Barlaud, P. Mathiew and I. Daubechies, "Image coding using wavelet transform", in IEEE Transactions on Image Processing, vol. 01, no. 02, Abril 1992.

[34]– S. Mallat, "Multiresolution Approximation and wavelets", in IEEE Transaction on Pattern Analysis and Machine Intelligence, vol.11, no. 07, July 1989.

[35]– I. Daubachies, "Orthogonal bases of compactly supported wavelets", in *Communications on Pure Applied Mathematics*, vol. XLI, pp.909-996, 1988.

[36]– M. Vetterli; C. Herley – " Wavelets and Filter Banks: Relationships and New Result" – Proc. IEEE ICASSP, Alburquerque, USA, Abril, 2000.

[37]– Y. Iano, L. R. Mendes, V. B. Sablón and J. A. Nalon, "Televisão digital de alta definição – HDTV", in Revista do Instituto Nacional de Telecomunicações, Brasil, vol.03, no.01, pp.1-9, April, 2000.

[38]– I. Daubechies – "Orthogonal Bases of Compactly Supported Wavelets II – Variations on a Theme" – ATT&Bell Lab., Tech. Report TM 11217-89111617, USA, 1990.

[39]– A. Said and W. A. Pearlman – "Image Compression Using The Spatial-Orientation Tree" – *IEEE International Symposium on Circuits and Systems*, Chicago, IL, USA, pp. 279-282, Maio, 1993.

[40]– A. Said and W. A. Pearlman – "A New, Fast and Efficient Image Coded Based on Set Partitioning in Hierarchical Trees" –. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 243-250, Junho, 1996.

[41]– P. N. Topiwala – "Wavelet Image and Video Compression" – Ed. Kluwer Academic Publishers, Massachusetts, USA, 1998.

[42]– J. M. Shapiro – "Embedded Image Coding Using Zerotrees of Wavelet Coefficients" – *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3445-3462, Dezembro, 1993.

[43]– "Handbook of Image and Video processing" – Ed. Al Bovik, Academic Press, San Diego, USA, 2000.

[44]– K. R. Rao nd P. C. Yip – "The Transform and Data Compression Handbook" – Ed. CRC Press, LLC, USA, 2001.

[45]– K. Sayood – "Introduction to Data Compression" – 2nd ed, Ed. Morgan Kaufmann, CA, USA, 2000.

[46]– J. Tian and R. O. Wells Jr – "A Lossy Image Codec Based on Index Coding" – *IEEE Data Compression Conference*, DCC'96, pp. 456, 1996.

[47]– J. Tian and R. O. Wells Jr – "Image Data Processing in the Compressed Wavelet Domain" – *3rd International Conference on Signal Processing Proc.*, pp. 978-981, Beijing, China, 1996.

[48]– J. Tian and R. O. Wells Jr – "Embedded Image Coding Using Wavelet-Difference-Reduction" – Wavelet Image and Video Compression, pp. 289-301, Kluwer Academic, Norwell, MA, USA, 1998.

[49]– J. S. Walker – "A Lossy Image Codec Based on Adaptatively Scanned Wavelet Difference Reduction" – Optical Engineering, in press.

[50]– V. Baskharan and K. Konstantinides, "Image and video compression standards", Ed. Kluwer Academic Publisher, 1996.

[51]– International Organization for Standard/ International Electrotechnical commission, "MPEG-2 Video", Doc ISO/IEC IS 13818-2, ITU-T H.262, May 1996.

[52]– A. L. Silva, "Procedimentos para a redução de efeito de bloco em sinais de vídeo codificados por MPEG-2 Test Model 5", Dissertação de Mestrado, FEEC/UNICAMP, Março, 2001.

[53]– Y. Iano, V. B. Sablón, R.T. D. Pietro and L. R. Mendes,"Subsistema de compressão e codificação do sinal de video (Parte II)", in Revista do Instituto Nacional de Telecomunicações, vol.03, no.01, pp.19-27, Abril, 2000.

[54]– F. S. Silva, "Procedimentos para medição e minimização do efeito de bloco decorrente do processamento digital de imagens (PDI)", Dissertação de Mestrado, FEEC/UNICAMP, Fevereiro, 2001.

[55]– http://www.mpeg.org/MPEG/MSSG

[56]– E. Petajan, "The HDTV Grand Alliance System", in Proceeding of IEEE, USA, vol.83, no.7, pp.1094-1105, Julho, 1995.

[57]– European Telecommunication Standard ETS 200744, "Digital Video Broadcasting (DVB); Frame Structure, Channel Coding and Modulation for Digital Terrestrial Television (DVB-T)", ETSI, Março 1997.

[58]– ITU Doc 11A/Jxx-E, "Proposed draft new recommendation channel coding, frame structure and modulation scheme for terrestrial integrated services digital broadcasting (ISDB-T)", March 1999.

[59]– C. Chrysafis and A. Ortega, "Line Based, Reduced Memory, Wavelet Image Compression", IEEE Transaction on Image Processing, vol.9, no.3, pp. 378-389, Março 2000.

[60]– C. Chrysafis and A. Ortega, "Efficient Context-Based Entropy-Coding for Lossy Wavelet Image Compression", in Proc. IEEE Data Compression Conf., Snowbird, UT, pp. 241-250, 1997.

[61]– R. W. Buccigrossi and E. P. Simoncelli, "Image Compression via Joint Statistical Characterization in the Wavelet Domain", in IEEE Transactions on Image Processing, vol.8, no.12, pp.1688-1701, Dezembro, 1999.

[62]– M. Ohta and S. Nakagi, "Hybrid Picture Coding with Wavelet Transform and Overlapped Motion-Compensated Interframe Prediction Coding", in IEEE Transactions on Signal Processing, vol.41, no.12, pp.3416-3424, Dezembro, 1993.

[63]– J. Vass, B. –B. Chai, K. Palaniappan and X. Zhuang, "Significance-Linked Connected Component Analysis for Very Low Bit-Rate Wavelet Video Coding", in IEEE Transactions on Cyrcuits and Systems for Video Technology, vol.9 no.4, pp. 630-647, Junho 1999.

[64]– X. Yang and K. Ramchandran, "Scalable Wavelet Video Coding Using Aliasing-Reduced Hierarchical Motion Compensation", in IEEE Transactions on Image Processing, vol.9 no.5, pp. 778-791, Maio, 2000.

[65]– F. A. Mujica, J. –P. Leduc, R. Murenzi and M.J.T. Smith, "A New Motion Parameters Estimation Algorithm Based on the Continuous Wavelet Transform", in IEEE Transactions on Image Processing, vol.9 no.5, pp. 873-888, Maio, 2000.

[66]– H. -W. Park and H.-S. Kim, "Motion Estimation Using Low-Band-Shift Method for Wavelet-Based Moving Picture Coding", in IEEE Transactions on Image Processing, vol.9 no.4, pp. 577-587, Abril 2000.

[67]– M. Rabbani and P. W. Jones, "Digital Image Compression Techniques", SPIE Optical Engineering Press, 1991.

[68]– J. Li and C.-C. J. Kuo, "Image compression with a hybrid wavelet-fractal coder" IEEE Trans. Image Processing, vol. 8, no. 6, pp. 868–874, Jun. 1999

[69]– C. –M Lai, K. –M. Lam and W. –C. Siu, "Fast fractal image coding based on kick-out and zero contrast conditions", IEEE Trans. Image Proc, vol. 12, no.11, pp. 1398–1403, Nov. 2003

[70]– T. K. Truong, J. H. Jeng, I. S. Reed, P. C. Lee and A. Q. Li, "A fast encoding algorithm for fractal image compression using the DCT inner product", IEEE Trans. Image Proc., vol. 9, pp. 529–535, Ap. 2000.

Apêndice A

Códigos-fonte desenvolvidos na pesquisa

Neste apêndice são mostradas as partes principais dos códigos desenvolvidos em Matlab 6.1 durante a pesquisa.

A.1 – Função de codificação SPIHT:

```
function [bitstream, n0, bpp, tempo] = SPIHTcoder(imgsaida, S, passos, bitbudget, verbose)

% uso: [bitstream,n0,bpp,tempo] = SPIHTcoder(imgsaida, S, passos, bitbudget)
% Codificador SPIHT para imagem formada por coeficientes-wavelet (baseado em SPIHT do livro de Sayood).
% Foi tb usada a varredura diferente para subimagens H, V e D de acordo com pg 281 do livro de Rao e Yip.
% Parâmetros de Entrada:
%  imgsaida:  Coef-wavelet que devem estar no formato imagem saido de C2imgsaida
%  S:         Matriz de tamanhos das subimagens descrita em decwaveletv2.
%  passos:    Numero de passos de SPIHT desejado
%  bitbudget: Limite de bits por pixel desejado. O codificador para quando atinge Passos
%             ou bitbudget, o que acontecer primeiro.
%  verbose:   Variavel que indica se o acompanhamento de codificaçao deve ser mostrado
% Parâmetros de Saida:
%  bitstream: Sequencia binaria sem compressao lossless.
%  n0:        Valor para fazer o threshold inicial.
%  bpp:       Valor da taxa de bits estimada usando entropia binaria
%  tempo:     Tempo em segundos gasto na codificação

% conversao do range dos coeficientes (entre os coef-wavelet de 0 a 1 e os valores de Threshold)

imgsaida = floor(.5+imgsaida*255);
tempo = cputime;

% inicializaçao de variaveis
bitstream = '';
nivdec = length(S(:, 1))-2;
n0 = ceil(log2(max(max(imgsaida))))-1;          % det. n0 a partir do maior coeficiente-wavelet
n=n0; T = 2^n;                                  % determina n e o Threshold inicial
nbits_max = round(bitbudget * prod(S(nivdec+2,:)));

% inicializaçao das listas  (Listas de Sayood - SPIHT)
LIP= varrSPIHT(S(1,:), 1)';
LIS= zeros(0,2);
for i=1:length(LIP(:,1))
   if not(all(mod(LIP(i,:),2)))
```

```
    LIS = [LIS;LIP(i,:)];
  end
end
LSP = zeros(0,2);
prox_LIP= zeros (0,2);
prox_LIS= zeros (0,2);
prox_LSP= zeros (0,2);
nbits = 0;

%XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
%X                LOOP PRINCIPAL                X
%XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

for k=1:passos
  % PROCESSANDO A LIP
  for i=1:length(LIP(:,1))
    if (abs(imgsaida(LIP(i,1),LIP(i,2)))>=T)  % teste de significancia
      prox_LSP = [prox_LSP; LIP(i,:)];  % se maior/igual que T, manda 1
      bitstream =[bitstream, '1'];
      if (imgsaida(LIP(i,1),LIP(i,2))>=0)       % se positivo, manda novo 0
        bitstream =[bitstream, '0'];
      else
        bitstream =[bitstream, '1'];  % cc, manda 1
      end
      nbits = nbits+2;
    else
      prox_LIP = [prox_LIP; LIP(i,:)];
      bitstream =[bitstream, '0'];      % se menor que T, manda 0
      nbits = nbits +1;
    end
    if (nbits>nbits_max)
      break
    end
  end

  % PROCESSANDO A LIS
  i=1;
  while ((i<=length(LIS(:,1)))&(nbits<=nbits_max))
    desc = descendencia(S,LIS(i,:));
    if (any(abs(imgsaida(desc))>=T))        % teste de significancia
      bitstream = [bitstream, '1'];   % se maior que T, manda 1
      nbits = nbits+1;
      for j=1:4                    % analisando os filhos diretos do coef-pai
        if (abs(imgsaida(desc(j)))>=T)
          bitstream = [bitstream, '1'];      % se filho maior que T, manda 1
          prox_LSP = [prox_LSP; [mod(desc(j)-1,S(nivdec+2, 1)), floor((desc(j)-
1)/S(nivdec+2,1))]+1 ];
          if (imgsaida(desc(j))>=0)
            bitstream = [bitstream, '0'];              % se filho positivo, manda 0
```

```
            else
              bitstream = [bitstream, '1'];              % se filho negativo, manda 1
            end
            nbits = nbits+2;
          else
            bitstream = [bitstream, '0'];      % se filho menor que T, manda 0
            prox_LIP = [prox_LIP; [mod(desc(j)-1,S(nivdec+2, 1)), floor((desc(j)-
1)/S(nivdec+2,1))]+1 ];
            nbits = nbits+1;
          end
          if (length(desc)>4)
            LIS = [LIS; [mod(desc(j)-1,S(nivdec+2, 1)), floor((desc(j)-1)/S(nivdec+2,1))]+1 ]; %
atualiza desc. (netos)
                                                % para ser analisada ainda nesse passo.
          end
        end
      elseif (not(isempty(desc)))
        bitstream = [bitstream, '0'];  % se menor que T, manda 0
        prox_LIS = [prox_LIS; LIS(i,:)];% mantem o coef em LIS para o proximo passo
        nbits = nbits+1;
      end
      i=i+1;
    end

    % PROCESSANDO A LSP (PASSO DE REFINAMENTO): enviar o MSB dos ecof.
significativos
    i=1;
    while( (i<=length(LSP(:, 1))) & (nbits<=nbits_max))
      binario = dec2bin(imgsaida(LSP(i,1),LSP(i,2)), 16);
      MSB = binario(16-n);
      bitstream = [bitstream, MSB];              % acresce o MSB dos significativos nos bits
trasmitidos
      nbits = nbits+1; i=i+1;
    end

    if (nbits>nbits_max)
      break;
    end

    % ATUALIZANDO AS LISTAS e o par n e T PARA O PROXIMO PASSO
    LIP=prox_LIP;
    prox_LIP= zeros(0,2);
    LIS=prox_LIS;
    prox_LIS= zeros(0,2);
    LSP=prox_LSP;

    n=n-1;
    T = 2^n;
end
```

```
tempo = cputime-tempo;

% CALCULO DO NUMERO DE BITS
prob = mean(bitstream == '1');
Hbits = -prob*log2(prob)-(1-prob)*log2(1-prob);
nbits = ceil(Hbits*length(bitstream));

bpp = nbits/prod(S(nivdec+2,:));
buff1 = sprintf('numero de bits = %d (%6.4f bpp)', nbits, bpp);
buff2 = sprintf('tempo de codificaçao = %5.2f seg (%2dm%4.2fs)', tempo, floor(tempo/60),
mod(tempo,60) );

if exist('verbose')
   disp(buff1);
   disp(buff2);
end
```

A.2 – Função de decodificação SPIHT:

```
function [imgdecod,tempo] = SPIHTdecoder4_budg(bitstream, S, n0, verbose)

% uso: [imgdecod,tempo] = SPIHTdecoder4_budg(bitstream, S, n0, verbose)
% Decodificador SPIHT para bitstream gerado pelo codificador SPIHT-gemeo dele (baseado em
SPIHT do livro de Sayood).
% Para mais especificaçoes, vide comentarios do codificador (help SPIHTcoder)
% Parametros de Entrada :
%  S:       Matriz de tamanhos das subimagens descrita em decwaveletv2.
%  bitstream: Sequencia binaria sem compressao lossless.
%  n0:      Valor para fazer o threshold inicial.
%  verbose:  Variavel que indica se o acompanhamento de codificaçao deve ser mostrado
% Parametros de Saida :
%  imgdecod  Coef-wavelet no formato imagem saido de C2imgsaida resultado da decodificaçao
%  tempo:    Tempo usado na decodificaçao

tempo = cputime;
% inicializaçao de variaveis
nivdec = length(S(:, 1))-2;
n = n0; T = 2^n;                    % determina o valor de n e do Threshold inicial
imgdecod = zeros(S(nivdec+2, :));      % inicializaçao de imgdecod como zeros


% inicializaçao das listas  (Listas de Sayood - SPIHT)
LIP= varrSPIHT(S(1,:), 1)';
LIS= zeros(0,2);
for i=1:length(LIP(:,1))
   if not(all(mod(LIP(i,:),2)))
```

```
      LIS = [LIS;LIP(i,:)];
   end
end
LSP = zeros(0,2);
prox_LIP= zeros (0,2);
prox_LIS= zeros (0,2);

 %XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
%X                  LOOP PRINCIPAL                  X
%XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

comp = length(bitstream);
pos = 1;
while (pos <= comp)
   % PROCESSANDO A LIP
   for i=1:length(LIP(:,1))
      if (bitstream(pos)== '1')  % teste de significancia
         prox_LSP = [prox_LSP; LIP(i,:)];          % eh maior/igual que T.
         pos=pos+1;
         if (bitstream(pos)== '0')          % eh positivo
            imgdecod(LIP(i,1), LIP(i,2))=  1.5*T;       % reconstroi valor positivo
         else
            imgdecod(LIP(i,1), LIP(i,2))= - 1.5*T;       % reconstroi valor negativo
         end
      else
         prox_LIP = [prox_LIP; LIP(i,:)];          % eh menor que T. Continua nulo e vai para
prox_LIP.
      end
      if pos>=comp
         break
      end
      pos=pos+1;
   end

   % PROCESSANDO A LIS
   i=1;
   while ((i<=length(LIS(:,1)))& (pos<=comp))
      desc = descendencia(S,LIS(i,:));
      if (bitstream(pos)== '1')          % teste de significancia
         pos=pos+1;
         for j=1:4                        % analisando os filhos diretos do coef-pai
            if (bitstream(pos)== '1')       % se 1, filho eh significativo
               prox_LSP = [prox_LSP; [mod(desc(j)-1,S(nivdec+2, 1)), floor((desc(j)-
1)/S(nivdec+2,1))]+1 ];
               pos=pos+1;
               if (bitstream(pos)== '0')       % se 0, filho eh positivo
                  imgdecod(mod(desc(j)-1,S(nivdec+2, 1))+1, floor((desc(j)-1)/S(nivdec+2,1))+1)=
1.5*T;
```

```
            else                    % se 1, filho eh negativo
              imgdecod(mod(desc(j)-1,S(nivdec+2, 1))+1, floor((desc(j)-1)/S(nivdec+2,1))+1)= -
1.5*T;
            end
          else                      % se 0, filho nao eh significativo
            prox_LIP = [prox_LIP; [mod(desc(j)-1,S(nivdec+2, 1)), floor((desc(j)-
1)/S(nivdec+2,1))]+1 ];
          end
          if (length(desc)>4)   % testa se existem netos
            LIS = [LIS; [mod(desc(j)-1,S(nivdec+2, 1)), floor((desc(j)-1)/S(nivdec+2,1))]+1 ];%
atualiza desc.
          end                                   % (netos) para ser analisada ainda nesse passo.
          pos=pos+1;
        end
      elseif (not(isempty(desc)))
        prox_LIS = [prox_LIS; LIS(i,:)];    % mantem o coef em LIS para o proximo passo
        pos=pos+1;
      end
      i=i+1;
    end

    % PROCESSANDO A LSP (PASSO DE REFINAMENTO):
    i=1;
    while( (i<=length(LSP(:, 1))) & (pos<=comp))
      if (bitstream(pos)=='1')
        imgdecod(LSP(i,1),LSP(i,2))= imgdecod(LSP(i,1),LSP(i,2))+ T/2 ;
      else
        imgdecod(LSP(i,1),LSP(i,2))= imgdecod(LSP(i,1),LSP(i,2))- T/2 ;
      end
      pos = pos+1; i = i+1;
    end

    % ATUALIZANDO AS LISTAS e o par n e T PARA O PROXIMO PASSO
    LIP=prox_LIP;
    prox_LIP= zeros(0,2);
    LIS=prox_LIS;
    prox_LIS= zeros(0,2);
    LSP=prox_LSP;

    n=n-1;
    T = 2^n;
end

imgdecod = imgdecod/255;

tempo = cputime-tempo;
buff = sprintf('tempo de decodificaçao = %5.2f seg (%2dm%4.2fs)', tempo, floor(tempo/60),
mod(tempo,60) );
if exist('verbose')
```

```
   disp(buff);
end
```

A.3 – Função de codificação fractal QPIFS:

```
function [tocoder, partree] = fractcoder2(img, err_per, qdtree, overlap, bits_SO, verbose)

% EXP 1: Codificador Fractal de Imagens V2.0.
% Uso: [tocoder, partree] = fractcoder2(img, errper_max, qdtree, overlap, bits_SO).
% Parametros de entrada:
% img:      Variavel contendo a imagem a ser codificada ou o caminho do arquivo que contem
%           a imagem. A imagem deve ter dimensoes em potencia de 2;
% err_per:  Erro percentual maximo permitido entre um matching R-D em funçao da variancia da
imagem;
% qdtree:   Vetor contendo o minimo e o maxido nivel de quadtree permitido para os blocos
Range.
% overlap:  Percentual de overlap na construçao do Domain Pool.
% bits_SO:  vetor de 2 posiçoes contendo o numero de bits no qual serao quantizados os
parametros S e O
% Parametros de saida:
% tocoder:  contem a relaçao D-R e trans. afim de todos os blocos codificados em forma de
matriz
%           5xn : [Range, Domain, T, S, O].
% partree:  eh a arvore de particao quadtree descrito nos comments do codificador.

if ischar(img)
   im_ori = double(imread(img));
elseif isa(img,'uint8')
   im_ori = double(img);
elseif isa(img,'double')
   im_ori = img;
else
   error('A imagem de entrada deve ser numerica (Double ou Uint8) ou o nome do arquivo bitmap
valido');
end

% Inicializaçao de constantes
qdtree_min = qdtree(1);          % Menor nivel do Quadtree: VAR.
qdtree_max = qdtree(2);          % Maior nivel do Quadtree: VAR.

% Inicializaçao de variaveis
e_max = err_per * (mean2(im_ori.^2) - mean2(im_ori)^2);
tam = size(im_ori);              % Tamanho da imagem a ser codificada.
%qdtree_max = 7;                 % Maior nivel do Quadtree: VAR.
partree = ntree (4, qdtree_min);     % Definiçao da arvore de partiçao inicial.
                                 % Obs.: O Quadtre que sera utilizado ser´a o UP-DOWN.

% Calculo do numero de blocos domain para cada nivel de quadtree.
```

```
nb_domain = zeros (qdtree_max-qdtree_min +1 ,1);
for i = 0:(qdtree_max-qdtree_min)
   tam_dom = tam / (2^(qdtree_min -1 +i));
   desl = max( 2, round(tam_dom(1)*(1-overlap)) ); % tamanho do deslocamento
   nb_domain(i+1) = (1+floor( (tam(1) - tam_dom(1))/desl ))^2; % Numero de blocos domain
                                    %  (Cada linha refere-se a um nivel).
end
nbc_domain = [0; cumsum(nb_domain)];    % nbc_domain contem a posicao de inicio de cada
nivel dentro do domain_pool


% CRIAÇAO DA LISTA DE DOMAIN POOL: contem informaçao para localizar e separa cada
% bloco dominio
domain_pool = zeros(11, sum(nb_domain));
% Subamostrando imagem original para formar domain pool
im_dom = (im_ori(1:2:tam(1),:) + im_ori(2:2:tam(1),:))/2; % subamostrando linhas
im_dom = (im_dom(:,1:2:tam(2)) + im_dom(:,2:2:tam(2)))/2; % subamostrando colunas

for i =1:sum(nb_domain)
   % Calculo do tamanho do bloco dominio
   aux = min(find(nbc_domain >=i))-1;
   domain_pool(3:4,i) = tam' / ( 2^(qdtree_min - 1 + aux) );

   % Calculo do numero da linha e do numero da coluna do bloco dominio
   aux_pos = i - nbc_domain(aux);
   desl = max( 1, round(domain_pool(3,i)*(1-overlap)) );          % tamanho do deslocamento
   nbd_rowcol = 1+floor( (tam(1)/2 - domain_pool(3,i))/desl ) ;          % Calcula o numero de
blocos dominio em uma linha ou coluna
   domain_pool(1,i) = floor( (aux_pos -1) / nbd_rowcol ) * desl;
   domain_pool(2,i) = mod  ( (aux_pos -1) , nbd_rowcol ) * desl;
   domain_pool(5:11,i) = classifica (im_dom(domain_pool(1,i)+[1:domain_pool(3,i)],
domain_pool(2,i)+[1:domain_pool(4,i)]), 1);
end


% LOOP MAIS GERAL: "MORE RANGE CELLS ?" (a serem processadas).

pos = 1;   % Contador que indica o bloco range que sera processado. Quando o contador chegar no
        % fim do vetor leaves(partree), entao teremos processado todos os blocos-range.

tocoder = zeros(6,0);

acomp = 0;              % Variavel auxiliar para caompanhamento do porcesso de codificaçao
tcomp = cputime;        % Variavel auxiliar para acompanhamento do tempo de codificaçao

lvs = leaves(partree);                  % Determina todos os nos terminais da arvore
while (pos <= length(lvs))          % Enquanto ainda houver blocos a processar
% Seleciona a folha na arvore que sera processado e seus indices
```

```
   [nd_dep nd_pos] = ind2depo(4, lvs(pos));    % Calcula a Profundidade (depth) e a Posiçao (pos)
da folha a ser processada
   nd_tam = tam / (2^nd_dep);                  % Calcula o Tamanho do bloco (folha) a ser processado
   % Calculo da posiçao na imagem do bloco (folha) a ser processado em qualquer nivel (dinamico)
   [nd_row, nd_col] = calc_rowcol(nd_pos, nd_dep, nd_tam);
   rangeblock = im_ori(nd_row +[1:nd_tam(1)], nd_col+[1:nd_tam(2)]);  % Armazena o bloco-
imagem que sera processado

   rangeclass = classifica(rangeblock, 0);

   % Encontrando o conjunto de dominio com o dobro do tamanho do range a ser processado;
      % Encontrando os vetores de restriçoes
   aux1 = find(domain_pool( 3,:)== nd_tam(1));
   aux2 = find(domain_pool( 4,:)== nd_tam(2));
   aux3 = find(domain_pool( 7,:)== rangeclass(3)); % Comparacçao da classe de media
   aux4 = find(domain_pool( 8,:)== rangeclass(4)); % Comparaçao da classe de variancia (S
positivo)
   aux5 = find(domain_pool(10,:)== rangeclass(4)); % Comparaçao da classe de variancia (S
negativo)
      % Aplicando as restirçoes e determinando os candidatos
   aux6 = union(aux4,aux5);
   domains_idx = intersect(intersect(aux1,aux2), intersect(aux3,aux6));  % Classificaçao
%   domains_idx = intersect(aux1,aux2);  % Alteraçao para eliminar classificaçao
   domains = domain_pool( :, domains_idx );    % Seleciona todos os dominios candidatos

   % Garantindo que um bloco de ultima camada sempre tenha candidatos evitando que ele fique
sem codificaçao
      % Testa se ha candidatos para blocos de ultima nivel quadtree
   if ( ((length(domains_idx)==0) & (nd_dep == qdtree_max)) )
      domains_idx = intersect(intersect(aux1,aux2), aux3);    % Aplica apenas uma restriçao de
classe (a de media)
      domains = domain_pool( :, domains_idx );            % Reseleciona os dominios candidatos
   end
      % Testa se agora ha candiatos, se nao houver, remove mais uma restriçao
   if ( ((length(domains_idx)==0) & (nd_dep == qdtree_max)) )
      domains_idx = intersect(aux1,aux2);                 % Nao aplica nenhuma restriçao de classe
      domains = domain_pool( :, domains_idx );            % Reseleciona os dominios candidatos
   end

   clear aux*;

   bestfit = zeros(5,1); bestfit(5) = Inf;

   % Testando o Matching para todos os dominios selecionados
   for i=1:size(domains,2)
      % Selecionando o domainblock que sera testado
      domainblock = im_dom(domains(1,i)+[1:domains(3,i)], domains(2,i)+[1:domains(4,i)]);
      domainblock2 = domainblock;    % Cria uma copia para S negativo
```

```
    % Ha classificaçao, logo so ha 2 transformaçoes afins: uma para si positivo e outra para negativo
      % Calculando os valores de Si, Oi e mse para Si positivo
      Tr = calc_tr(rangeclass, domains(9,i) );
        % Aplicando a Transformaçao Afim
      if (Tr>3)
         domainblock = domainblock';
      end
      domainblock = rot90(domainblock, mod(Tr,4));
        % Calculando o bestfit
      [Si, Oi, mse] = calc_SOmse (rangeblock, domainblock, rangeclass(1), domains(5,i),
domains(6,i), bits_SO);
      if (mse < bestfit(5))
         bestfit = [ domains_idx(i); Tr; Si; Oi; mse];
      end

      % Calculando os valores de Si, Oi e mse para Si negativos
      Tr = calc_tr(rangeclass, domains(11,i) );
        % Aplicando a Transformaçao Afim
      if (Tr>3)
         domainblock2 = domainblock2';
      end
      domainblock2 = rot90(domainblock2, mod(Tr,4));
        % Calculando o bestfit
      [Si, Oi, mse] = calc_SOmse (rangeblock, domainblock2, rangeclass(1), domains(5,i),
domains(6,i), bits_SO);
      if (mse < bestfit(5))
         bestfit = [ domains_idx(i); Tr; Si; Oi; mse];
      end

      % Vai para o proximo dominio
    end

    % Foi encontrado pelo menos um matching dentro da tolerancia ou maximo nivel de quadtree
          %alcançado
    if ((bestfit(5) <= e_max) | (nd_dep == qdtree_max))
      % Foi encontrado um matching dentro da tolerancia
      tocoder = [ tocoder, [lvs(pos); bestfit(1:5)] ];   % Armazena os dados para o codif. lossless
      pos = pos +1;
      % Acompanhamento da evoluçao da codificaçao
      if ( (( (nd_pos+1)/4^nd_dep)>=acomp) & exist('verbose') )
         buf = sprintf('Concluidos: %4.1f %%', 100*acomp);
         disp(buf);
         acomp = acomp + 0.1;
      end
    else
      % Nao foi encontrado um matching dentro da tolerancia
      partree = nodesplt(partree, lvs(pos));  % Quebra a folha
      lvs = leaves(partree);
    end
```

```
end

tempo = cputime-tcomp;
if (exist('verbose'))
   buf = sprintf('Tempo Usado na Codificaçao: %8.3f s (%dm%4.2fs)', tempo, floor(tempo/60),
mod(tempo,60));
   disp(buf);
end
```

A.4 – Função de decodificação fractal QPIFS:

```
function imgdecod = fractdecoder2(tocoder, partree, tam_im, tol, qdtree, overlap)

% Uso: imgdecod = fractdecoder2(tocoder, partree, tam_im, tol, qdtree, overlap)
% Decodificador para codificador  fractal.
% Parametros:
% tocoder:    contem a relaçao D-R e trans. afim de todos os blocos codificados em forma de matriz
%             5xn : [Range, Domain, T, S, O].
% partree:    eh a arvore de particao quadtree descrito nos comments do codificador.
% tam_im:     referencia o tamanho no qual a imagem sera decodificada (deve ser potencia de 2);
% tol:        representa a tolerancia ao erro que determina a convergencia do decodificador;
% qdtree:     Vetor contendo o minimo e o maxido nivel de quadtree permitido para os blocos
Range.
% overlap :   percentual de overlap na construçao do Domain Pool.

% Inicializaçao de constantes
qdtree_min = qdtree(1);              % Menor nivel do Quadtree: VAR.
qdtree_max = qdtree(2);              % Maior nivel do Quadtree: VAR.

% Inicializaçao de variaveis
comp = length (tocoder(1, :)); % Calculo de n para fazer o contador
rearranjo = zeros(9,comp);     % Inicializaçao da variavel que rearranja as informaçoes de
tocoder.Converte a
                    % numeraçao qudtree (de domains e ranges)para coordenadas.
                  % para facilitar a decodificaçao.
                  % [ Posiçao do bloco-range na imagem (x, y),
                  %   Posiçao do bloco-dominio na imagem (x, y),
                  %   Tamanho do bloco-range (dx, dy),
                  %   Indice da transformaçao geometrica usada (de 0 a 7),
                  %   Ajuste de contraste Si
                  %   Ajuste de brilho Oi  ]
imgaux = zeros(tam_im);  % criando a img que gerarah imgdecod apos as iteraçoes
imgdecod = imgaux; % Inicalizaçao de imgdecod

% XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
% X                                          X
% X     REALIZANDO O REARRANJO DAS INFORMAÇOES DE  X
%X      TOCODER QUE FOI TRANSMITIDO          X
```

```
% X                                                                    X
% XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

% Calculo do numero de blocos domain para cada nivel de quadtree.
nb_domain = zeros (qdtree_max-qdtree_min +1 ,1);
for i = 0:(qdtree_max-qdtree_min)
   tam_dom = tam_im / (2^(qdtree_min -1 +i));
   desl = max( 2, round(tam_dom(1)*(1-overlap)) ); % tamanho do deslocamento
   nb_domain(i+1) = (1+floor( (tam_im(1) - tam_dom(1))/desl ))^2; % Numero de blocos domain
                                    %  (Cada linha refere-se a um nivel).
end
nbc_domain = [0; cumsum(nb_domain)];    % nbc_domain contem a posicao de inicio de cada
nivel de quadtree
                                % dentro do domain_pool
% CRIAÇAO DO DOMAIN-POOL
% CRIAÇAO DA LISTA DE DOMAIN POOL
% O domain-pool contem informaçao para localizar e separar cada bloco dominio
domain_pool = zeros(4, sum(nb_domain));    % inicializaçao. 4 informaçoes: 1)num. da linha, 2)
num. da coluna de
                                % cada bloco-domain candidato. 3 e 4) tamanho do bloco (dx, dy)
for i =1:sum(nb_domain)
   % Calculo do tamanho do bloco dominio
   aux = min(find(cumsum(nb_domain)>=i));
   domain_pool(3:4,i) = tam_im' / ( 2^(qdtree_min - 2 + aux) );

   % Calculo do numero da linha e do numero da coluna do bloco dominio
   aux_pos = i - nbc_domain(aux);
   desl = max( 2, round(domain_pool(3,i)*(1-overlap)) );          % tamanho do deslocamento
   nbd_rowcol = 1+floor( (tam_im(1) - domain_pool(3,i))/desl ) ;              % Calcula o numero
de blocos dominio em uma linha ou coluna
   domain_pool(1,i) = floor( (aux_pos -1) / nbd_rowcol ) * desl;
   domain_pool(2,i) = mod  ( (aux_pos -1) , nbd_rowcol ) * desl;
end

% REARRANJO EM SI PARA A SUBIMAGEM

for pos = 1:comp
   % Seleciona o bloco-range que sera processado e seus indices
   [nd_dep nd_pos] = ind2depo(4, tocoder(1,pos));   % Calcula a Profundidade (depth) e a Posiçao
(pos) da folha
   nd_tam = tam_im / (2^nd_dep);                % Calcula o Tamanho do bloco (folha) a ser
processado
   % Calculo da posiçao na imagem do bloco (folha) a ser processado em qualquer nivel (dinamico)
   aux = dec2bin(nd_pos, 2*nd_dep);
   nd_row = bin2dec( aux(1:2:(2*nd_dep)) ) * nd_tam(1);
   nd_col = bin2dec( aux(2:2:(2*nd_dep)) ) * nd_tam(2);

   %Executa o rearranjo
   rearranjo(1:2, pos) = [nd_row; nd_col];              % (x,y) do bloco-range
```

```
    rearranjo(3:4, pos) = domain_pool(1:2, tocoder(2, pos));  % (x,y) do bloco-domain gemeo
    rearranjo(5:6, pos) = nd_tam';                  % (dx,dy) do bloco-range
    rearranjo(7:9, pos) = tocoder (3:5, pos);          % transformaçao

end

% XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
% X                        ITERAÇOES FRACTAIS                    X
% XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
erro = inf;
iter = 1;
while ((erro > tol) & (iter <10))
   for i = 1:length(rearranjo(1, :))
      domainblock = imgdecod ( rearranjo(3,i)+[1:2*rearranjo(5,i)],
rearranjo(4,i)+[1:2*rearranjo(6,i)] );
      aux = (domainblock(1:2:2*rearranjo(5,i),:)+domainblock(2:2:2*rearranjo(5,i),:))/2;% Media
das linh(subamost)
      domainblock = (aux(:,1:2:2*rearranjo(6,i))+aux(:,2:2:2*rearranjo(6,i)))/2;      % Media das
col(subamost)

      domainblock = rearranjo (8, i)*domainblock + rearranjo(9,i); % aplicaçao da transformaçao
      if (rearranjo(7,i) >= 4)
         domainblock = domainblock';
      end
      domainblock = rot90(domainblock, mod(rearranjo(7,i), 4) );
      imgaux ( rearranjo(1,i)+[1:rearranjo(5,i)], rearranjo(2,i)+[1:rearranjo(6,i)] )= domainblock;
   end
   erro = sqrt(  sum(sum((imgdecod-imgaux).^2))/prod(size(imgdecod))  );
   imgdecod = imgaux;
   iter = iter+1;
end
```

A.5 – Função de codificação de vídeo proposta:

```
function FW_videocoder(sSeq, nFrames, nF0, rBPS, rFPS, sFile)

% Uso: txbits = FW_videocoder(sSeq, nFrames, nF0, rBPS, rFPS, sFile);
%  Funçao Codificador de Video Fracal-Wavelet (Tese Doutorado Fernando)
%  Parametros:
%     sSeq:   Nome da sequenca de video que sera codificada
%     nFrames: Numero de frames a ser codificado
%     nF0:    Primeiro frame a ser codificado
%     rBPS:   Taxa de Bits Por Segundo almejada para a sequencia codificada
%     rFPS:   Numero de frames por segundo da sequencia
%     sFile:   Arquivo para armazenamento dos dados codificados
% OBSERVAÇOES:
%  A) A sequencia de entrada deve ter o tamanho de 480 linhas x 720 colunas.
```

```
% Setando parametros do codificador
nQdtree = [4 6];   % qdtree_min=4 e qdtree_max=6.
rOverlap = 1.00;   % overlap de blocos dominio.
nBitsSO = [5,5];   % numero de bits para quantizar S e O.
nTamGOP = 4;       % numero de frames por GOP
nBits = 0;         % contador de bits gastos
nNivDec = 3;       % numero de niveis de decomposiçao wavelet
nTJMax = [8 8];    % tamanho maximo da janela de procura da Estimaçao de Movimento
aTocoder = [];
nCompTc = [0];
nVectors =[];
bStream = '';
nCompStr=[0;0];
rOverhead = [];
nCompOver = [0];
nPB = 0;
sDwtStatus = dwtmode('status','nodisp');
dwtmode('per','nodisp');

rFramesRef=zeros(480,720,nFrames);
% Calculando parametros adicionais
nIP= 4;    % Define quantas Quadros P vale uma Quadro I
rTaxaMin = rBPS / (480*720*rFPS*(1+ (nIP-1)/nTamGOP));

% LOOP MAIS GERAL - MORE FRAMES
for nFrm = 0:(nFrames-1)
   disp(sprintf('Frame n°%d',nFrm+1));
   % Realizando a primeira leitura
   rImgNew = readppm(sprintf('%s%03d.ppm', sSeq, nFrm+nF0));
   nTam = size(rImgNew);
   % convertendo em P&B
   if length(nTam)==3
      rImgNew =rgb2gray(rImgNew);
      nTam = size(rImgNew);
      bColor = uint8(1);
   else
      bColor = uint8(0);
   end
   if any(nTam~=[480,720])
      error('A sequencia de entrada deve ter 480 linhas por 720 colunas');
   end

   % Aplicando a DWT e separando a Passa baixas
   [rCoef,rSizes]=wavedec2(rImgNew, 3, 'bior4.4');
   rDwtNew = c2img(rCoef,rSizes);
   rPB = 255 * rDwtNew(1:rSizes(1,1),1:rSizes(1,2))/(2^nNivDec);      % NivDec = 3 sempre
   nVecDet = [];      % Nao ha vetores de movimento

   % Verificando se a imagem e Quadro I
```

```
    if (~mod(nFrm, nTamGOP))
      % PROCESSANDO UMA QUADRO I
      % Processando a PB
      rErrPer = 0.0003 / (nIP*rTaxaMin);
      rErrMax = rErrPer * (mean2(rPB.^2) - mean2(rPB)^2);
      %   Codificaçao Fractal
      [aTcPB, tPtPB] = Intra_Fcod( rPB, rErrMax, nQdtree, rOverlap, nBitsSO, 'v');
      %   Escontrando os matchings ruins
      nNonMatches = find(aTcPB(6,:)>2.5*rErrMax);
      aTcPB(3,nNonMatches)=8; aTcPB(4,nNonMatches)=0; aTcPB(5,nNonMatches)=0;
      nMatches = setdiff( 1:size(aTcPB,2), nNonMatches);
      %   Codificaçao Lossless nao implementada. Estimaçao por entropia de primeira ordem
      nBitsFractal = calc_nbits(aTcPB(:,nMatches), tPtPB, nQdtree(2), nBitsSO);
      nBits = nBits + nBitsFractal + 8*length(nNonMatches);

      % Codificaçao das subimagens de DETALHES
      %   Calculando os parametros
      rTaxa = 1.3 * max(0.01, nIP*rTaxaMin - nBitsFractal/(480*720));
      LIP = zeros(0,2);
      rImg2Spiht = rDwtNew; rImg2Spiht(1:rSizes(1,1),1:rSizes(1,2))=0;
      %   Executando a codificaçao SPIHT
      [bBitStream, nN0, nNf] = Intra_SPIHTcod(rImg2Spiht, rSizes, 9, rTaxa , LIP);
      %   Estimando a entropia de primeira ordem
      rProb=mean(bBitStream=='1');
      nBits = nBits + (length(bBitStream)*(-rProb*log2(rProb)-(1-rProb)*log2(1-rProb)));

      % Calculando o Overhead da PB
      rPBPadded = zeros(2^max(ceil(log2(size(rPB)))));
      rPBPadded (1:rSizes(1,1),1:rSizes(1,2))=rPB;
      for i=1:length(nNonMatches)
        [nNdDep nNdPos] = ind2depo( 4, aTcPB(1,nNonMatches(i)) );   % Calcula a Profundidade
(depth) e a Posiçao (pos) da folha a ser processada
        nNdTam = size(rPBPadded) / (2^nNdDep);
        aux = dec2bin(nNdPos, 2*nNdDep);
        nNdRow = bin2dec( aux(1:2:(2*nNdDep)) ) * nNdTam(1);
        nNdCol = bin2dec( aux(2:2:(2*nNdDep)) ) * nNdTam(2);
        nVarr = varrspiht(nNdTam, 1);
        nVarr = ( nVarr(2,:)+nNdCol-1)*size(rPBPadded,1) + (nVarr(1,:)+nNdRow);
        rOverhead = [rOverhead, rPBPadded(nVarr)];
      end
      nCompOver = [nCompOver, length(rOverhead)];
      nPB = nPB + 1;
      % Atualizando a imagem referencia para as Quadros P
      LIP = zeros(0,2);
      rPBdec = Intra_Fdec(aTcPB, tPtPB, rOverhead((nCompOver(nPB)+1):nCompOver(nPB+1)),
rSizes(1,:), .0001, nQdtree, rOverlap);
      rDwtRef = Intra_SPIHTdec(bBitStream, rSizes, nN0, LIP);
      rDwtRef(1:rSizes(1,1),1:rSizes(1,2)) = (2^nNivDec)*rPBdec/255;
```

```
      aTocoder = [aTocoder, aTcPB];
      nCompTc = [nCompTc, size(aTocoder,2)];
   else
      % PROCESSANDO UMA QUADRO P
      % Processando a PB
      %   Codificaçao Lossless nao implementada. Estimaçao por entropia de primeira ordem
      % Executando a MRME e calculando os parametros
      [nVecDet, rDwtDiff] = MRMotionEst(rDwtNew, rDwtRef, rSizes, nTJMax);
      %   Executando a codificaçao SPIHT
      LIP = varrSPIHT(rSizes(1,:), 1)';
      [bBitStream, nN0, nNf] = Intra_SPIHTcod(rDwtDiff, rSizes, 9, 1.3*rTaxaMin , LIP);
      % Atualizando o Frame Referencia
      rDwtDif2 = Intra_SPIHTdec(bBitStream, rSizes, nN0, LIP);
      rDwtRef = MRMotionComp(rDwtDif2, rDwtRef, nVecDet, rSizes, nTJMax);
      %   Estimando a entropia de primeira ordem
      rProb=mean(bBitStream=='1');
      nBits = nBits + (length(bBitStream)*(-rProb*log2(rProb)-(1-rProb)*log2(1-rProb)));
   end
   % Atualizando informaçoes  que serao mandadas para o codificador lossless
aC = img2c(rDwtRef,rSizes);aa =  waverec2(aC,rSizes, 'bior4.4');rFramesRef(:,:,nFrm+1)=aa;
   bStream = strcat(bStream, bBitStream);
   nCompStr = [nCompStr, [length(bStream); nN0]];
   nVectors = [nVectors, nVecDet];
end
dwtmode(sDwtStatus,'nodisp');
txbits = round(nBits*rFPS/nFrames);
disp (sprintf('Total de bits: %d bits;  Taxa de bits/s: %d bps', nBits, txbits ));
eval(sprintf('save %s aTocoder nCompTc bStream nCompStr rOverhead nCompOver nVectors
rSizes nBits', sFile));
```

A.6 – Função de codificação de vídeo proposta:

```
function rFrames = FW_videodecoder(sFile, nFrames_max)

% Uso: rFrames = FW_videodecoder(sFile, nFrames_max)
%  Funçao Decodificador de Video Fracal-Wavelet (Tese Doutorado Fernando)
%  Parametros:
%     sFile:   Nome do arquivo que contém a sequencia codificada
%     nFrames_max: Numero máximo de frames a ser decodificado
%  Saida:
%     rFrames: frames decodificados

% Carregando o arquivo de parametros de entrada
eval(sprintf('load %s % aTocoder nCompTc bStream nCompStr rOverhead nCompOver nVectors
rSize', sFile));
```

```
% Configurando os parametros do codificador
nQdtree = [4 6];   % qdtree_min=4 e qdtree_max=6.
rOverlap = 1.00;   % overlap de blocos dominio.
nBitsSO = [5,5];   % numero de bits para quantizar S e O.
nTamGOP = 4;       % numero de frames por GOP
nNivDec = 3;       % numero de niveis de decomposiçao wavelet
nTJMax = [8 8];    % tamanho maximo da janela de procura da Estimaçao de Movimento
nVetPos = 0;
nPB = 1;
tPtPB = ntree(4,nQdtree(1));
sDwtStatus = dwtmode('status','nodisp');
dwtmode('per','nodisp');
% Calculando parametros adicionais
nFrames = min(length(nCompStr)-1,nFrames_max);
rFrames = zeros(480, 720, nFrames );

% LOOP MAIS GERAL - MORE FRAMES
for nFrm = 0:(nFrames-1)
  disp(sprintf('Frame n°%d',nFrm+1));
  % Separando o BitStream adequado
  bBitStream = bStream( (nCompStr(1,nFrm+1)+1):nCompStr(1,nFrm+2) );
  nN0 = nCompStr(2,nFrm+2);
  % Verificando se a imagem e Quadro I
  if (~mod(nFrm, nTamGOP))
    % PROCESSANDO UMA QUADRO I
    %  Processando a PB
    aTcPB = aTocoder(:, (nCompTc(nPB)+1):nCompTc(nPB+1) );
    rOver = rOverhead(:, (nCompOver(nPB)+1):nCompOver(nPB+1) );
    rPBdec = Intra_Fdec(aTcPB, tPtPB, rOver, rSizes(1,:), .0001, nQdtree, rOverlap);
    % Atualizando a imagem referencia para as Quadros P
    LIP = zeros(0,2);
    rDwtRec = Intra_SPIHTdec(bBitStream, rSizes, nN0, LIP);
    rDwtRec(1:rSizes(1,1),1:rSizes(1,2)) = (2^nNivDec)*rPBdec/255;
    rDwtRef = rDwtRec;
    nPB=nPB+1;
  else
    % PROCESSANDO UMA QUADRO P
    nVecDet = nVectors(:,nVetPos*13500 + [1:13500]);
    nVetPos=nVetPos+1;
    LIP = varrSPIHT(rSizes(1,:), 1)';
    rDwtDiff = Intra_SPIHTdec(bBitStream, rSizes, nN0, LIP);
    %  Executando a MRMC e calculando os parametros
    rDwtRec = MRMotionComp(rDwtDiff, rDwtRef, nVecDet, rSizes, nTJMax);
    %  Atualizando o Frame Referencia
    rDwtRef = rDwtRec;
  end
  % Atualizando os frames
  rCoef = img2c(rDwtRec,rSizes);
  rImgRec =  waverec2(rCoef,rSizes, 'bior4.4');
```

```
   rFrames(:,:,nFrm+1)= rImgRec;
end
dwtmode(sDwtStatus,'nodisp');
```

Lista de Publicações

09/2004 **Artigo :** "A Fast and Efficient Hybrid Fractal-*Wavelet* Image Coder"
Autores: Y. Iano, A.L.M.Cruz, F. S. Silva.
Tema: Fractais, Codificação de Imagens Digitais, Transformadas *Wavelet*, Velocidade de Processamento.
Aprovado pela revista *IEEE Transactions on Image Processing*, aprovado e a ser publicado na edição de Janeiro, 2006.

12/2004 **Artigo**: "A New Approach to Reduce Blocking Artifacts in MPEG-2 Video Coding"
Autores: Y. Iano, A.L. M.Cruz, F. S. Silva.
Tema: Processamento Digital de Vídeo: Redução de blocagem em codificadores MPEG-2, Qualidade de vídeo.
Publicado na Revista *Ciência e Tecnologia* UNISAL Ano VII – no. 11, 2004.

10/2004 **Artigo**: "A Simple Local Method to Reduce Blocking Effect"
Autores: Y. Iano, A.L.M.Cruz, F. S. Silva.
Tema: Processamento Digital, Redução de blocagem, Qualidade de Imagem.
Publicado na Revista *Ciência e Tecnologia* UNISAL Ano VIII – no. 12, 2005.

03/2005 **Artigo :** "Accelerating Fractal Image Coding Using Wavelet Transform"
Autores: Y. Iano, A.L.M.Cruz, F. S. Silva.
Tema: Fractais, Codificação Híbrida de Imagens Digitais, Transformadas *Wavelet*, Velocidade de Processamento.
Submetido à revista *IEEE Transactions on Image Processing.*