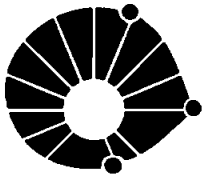
UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE COMUNICAÇÕES



TESE DE DOUTORADO

PROCEDIMENTOS PARA MÉTODO HÍBRIDO DE COMPRESSÃO DE IMAGENS DIGITAIS UTILIZANDO TRANSFORMADAS *WAVELET* E CODIFICAÇÃO FRACTAL.

Ana Lúcia Mendes Cruz Silvestre da Silva

Orientador: Prof. Dr. Yuzo Iano

Banca Examinadora:

Prof. Dr. David Fernandes (ITA)

Prof. Dr. Antônio Cláudio Paschoarelli Veiga (UFU)

Prof. Dr. Akebo Yakamani (FEEC - UNICAMP)

Prof. Dr. Max Henrique Machado Costa (FEEC - UNICAMP)

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

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

Campinas, SP – Brasil

Maio– 2005

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

Si38p

Silva, Ana Lúcia Mendes Cruz Silvestre da
Procedimentos para método híbrido de compressão de imagens digitais utilizando transformadas Wavelet e codificação fractal / Ana Lúcia Mendes Cruz 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 hybrid compression of digital images using wavelet transform and fractal coding
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: David Fernandes, Antônio Cláudio Paschoarelli Veiga, Akebo Yakamani, Max Henrique Machado Costa e Vicente Idalberto Becerra Sablón
Data da defesa: 05/05/2005

RESUMO

O principal obstáculo nas implementações da compressão fractal de images é o exaustivo tempo de codificação inerente. O objetivo desta pesquisa é introduzir uma nova aproximação para acelerar a codificação fractal de imagens através da aplicação da TWD e suas propriedades, sem que haja detrimento do PSNR ou da qualidade visual subjetiva. Logo, este trabalho apresenta um novo Codificador Híbrido Fractal-*Wavelet*, que aplica a compressão fractal acelerada à imagens estáticas decompostas pela transformada *wavelet*, explorando a correlação direcional das subimagens-*wavelet*. Este tipo de correlação foi constatada por Shapiro em outro contexto [2]. O esquema proposto promove melhor qualidade visual (compatível com as medidas de PSNR) e uma redução média de cerca de 80% no tempo de codificação-decodificação quando comparado aos resultados da codificação fractal pura para diversas imagens e taxas de bits. Adicionalmente os detalhes da imagem e as características de transmissão progressiva *wavelet* foram preservados. Nenhum artefato de blocagem, usualmente encontrados em codificadores fractais puros, resultou do processo de compressão híbrido. Os resultados deste trabalho demonstram o potencial da compressão híbrida fractal-*wavelet* como sendo uma ferramenta poderosa ainda a ser explorada.

ABSTRACT

The major drawback in the implementations of the fractal image compression is the exhaustive inherent encoding time. The objective of this research is to introduce a new approach to accelerate the fractal image coding through the application of the DWT and its properties without decrease in the PSNR as well as in the subjective visual quality. Thus, this work presents a New Fast Hybrid Fractal-Wavelet Image Coder that applies the accelerated fractal compression to wavelet transformed images by exploiting the directional correlation of the wavelet subimages. This kind of correlation was noticed by Shapiro in a different context [2]. The proposed scheme promotes better visual quality (compatible to the PSNR measures) and an average reduction of about 80% in encoding-decoding time when compared to the results of the pure accelerated fractal coding for several images and bitrates. Furthermore, the image details and the characteristics of wavelet progressive transmission are maintained; blocking effects, usually found in pure fractal coders, are not introduced. The results of this work demonstrate the potential of the fractal-wavelet hybrid compression as a powerful tool to be further explored.

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*

"Ele Me invocará, e Eu lhe responderei; estarei com ele na angústia; livrá-lo-ei e o glorificarei.
Salmos, 91:15

A Deus eu ofereço este trabalho feito com Esforço, Dedicação e muito Amor. Essa oração na qual busquei tanto apoio nos momentos difíceis, Te ofereço novamente agora, Senhor, como agradecimento por Ter estado ao meu lado nas alegrias e dificuldades.

Obrigada por cada pessoa que Colocastes em meu caminho e que, de alguma forma, me ajudaram, contribuindo para que esse trabalho se realizasse.

"Proponha-se a ser feliz! Fique atento a toda e qualquer possibilidade. Por menor que seja, agarre-a! É essa a oportunidade de ser e de se fazer feliz." *Darclê, Do Céu e da Terra*

Ofereço este trabalho também ao meu pai, Ary, definitivamente a pessoa mais corajosa que eu já conheci. Eu te amo muito, pai. Obrigada por ter me ensinado que a felicidade encontra-se definitivamente dentro de nós mesmos. À você, todo o meu amor. Agradeço todos os dias a honra e o privilégio de ser sua filha.

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 esta pesquisadora através dos Projetos: "**Algoritmos Computacionais para a Estimação de Movimento no Método Híbrido de Compressão em Ambiente MPEG-2 (PDI)**", Processo FAPESP no. 98/05321-8; "**Procedimentos para o Método Híbrido de Compressão em Processamento Digital de Imagem, com Destaque para Etapa de Compressão Espacial – (PDI)",** Processo FAPESP no. 98/12897-3 e "**Procedimentos para o Método Híbrido de Compressão em Processamento Digital, aplicando Transformada *Wavelet* – (PDI)",** Processo FAPESP Nº 00/11156-1 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

Ao alcançar o fim dessa etapa tão importante, gostaria de expressar meus agradecimentos às pessoas e instituições que tanto contribuíram direta ou indiretamente para a realização deste trabalho.

Primeiramente, desejo agradecer ao grande amigo e Professor Doutor Yuzo Iano que, como orientador, esteve sempre atento aos nossos questionamentos, estimulando-nos sempre a buscar novas idéias e soluções. Obrigado por ter me guiado desde a Iniciação Científica e, acima de tudo, por ter sido um grande amigo e um modelo de força, perseverança e serenidade. Foi um longo caminho este que percorremos juntos e, por sua imensa ajuda, eu agradeço. Obrigada especialmente por me ensinar que devemos ter serenidade mesmo nos momentos mais difíceis que se apresentam diante de nós.

À Universidade Estadual de Campinas (UNICAMP) pela oportunidade e ao apoio recebido durante a realização desta pesquisa. À FAPESP pelo suporte financeiro concedido a mim desde a Iniciação Científica e, sem o qual, este trabalho não estaria hoje concretizado.

Aos meus professores da Engenharia Elétrica da Unicamp que tanto contribuíram para a minha formação. Agradeço pelas preciosas lições e ensinamentos.

Ao meu querido pai, que têm se mostrado uma fortaleza de coragem diante das dificuldades, encontrando felicidade onde eu jamais achei que seria possível. Sua força, sua doçura e seu amor em face dos problemas são algo que jamais me esquecerei. Obrigada por ter me ensinado que a felicidade encontra-se definitivamente dentro de nós mesmos.

À minha querida mãe, obrigada por sua doçura e especialmente pela sua alegria tão imensa que nos contagia a todos. Obrigada por essa sua incrível capacidade de nos iluminar e por nos ajudar a seguir os caminhos de nossa bem-aventurança. Só a tua mera presença já nos alegra o coração. Agradeço por me ensinar especialmente que não há idade para realizarmos nossos sonhos.

A vocês, meus pais, todo o meu amor e admiração por terem sido ponto de referência e apoio em todos esses anos, estando sempre presentes mesmo à distância nos momentos de alegrias e dificuldades. Obrigada pelas orações. Muito me orgulha ter tido o privilégio de ter vocês como pais.

À família de meu marido Fernando, pelas palavras de apoio e pela compreensão com relação à nossa ausência em momentos dos quais gostaríamos de ter participado.

Ao querido amigo Martinez, pelas divertidas conversas, responsáveis por tantas alegrias e sorrisos. À você e à querida Dar e ao Sr Emílio, agradeço pelas palavras encorajadoras e tranqüilizadoras sempre presentes. À querida Dar e à Dona Sima eu agradeço especialmente por tudo que fizeram pelo meu pai e pelas orações, das quais nunca me esquecerei.

À minha avó, de quem todos sentimos saudades e de quem guardamos lembranças de muita alegria.

A Edinho, meu irmão querido, de quem eu sentia muitas saudades e que voltou para a nossa companhia. Agradeço por suas palavras tranqüilizadoras e por sua ajuda num momento em que eu precisava

de um sinal. Às queridas Tia Lu e Raphaela, pelas alegres visitas, pelas palavras encorajadoras e pelos mimos.

Aos amigos Sidney, Christina, Fabinho, Karina, Daniela, Gil e à todo o pessoal da Acqua com quem eu tanto aprendi. Agradeço o apoio, pelas pequenas vitórias cotidianas e por me ajudarem a sempre tentar superar os meus próprios limites.

À querida Didi por me ter ensinado a importância de sermos alegres para nós mesmos e para o mundo. A Uther por ter me "carregado" em momentos muito difíceis e por ter me "empurrado" nos momentos extremos em que, por muito pouco, não me cessaram as forças. Agradecimentos especiais também à Dan Raylock e a Ktur.

Ao meu padrinho Ju, de quem sinto muitas saudades. À Má, minha madrinha e querida Scarlet, todo o sucesso deste mundo.

Aos colegas de departamento e, em especial, ao amigo Vicente pela ajuda sempre presente e de boa vontade. Aos amigos muito queridos Washington, Warley, Noêmia e Lúcia sempre eficientes e que com sua simpatia nos ajudaram sempre desde a graduação. Agradecimentos especiais também à Gerusa, Giane, Mazé, Wilma e Érgio. Que vocês recebam em dobro toda a alegria cotidiana com que sempre nos atendem.

À DMB e à SFI, que me mantiveram ouvindo a música e que me ensinaram a olhar sempre para o horizonte. À minha música, aos meus desenhos e ao meu piano.

Ao meu querido marido Fernando, a pessoa mais generosa que já conheci e, sem dúvida nenhuma, a mais brilhante. A você todo o meu amor e gratidão. Agradeço por todo o amor, carinho, paciência e dedicação cotidianas, sem os quais eu certamente teria sucumbido em muitos momentos. A ele agradeço também pela ajuda, sugestões e palavras encorajadoras fundamentais para que esse trabalho fosse realizado. Agradeço por você ter sido sempre o meu porto seguro.

À Deus, pela iluminação, pela boa sorte e pelo trabalho. Por me mostrar que, mesmo nos momentos difíceis, sempre existem portas a serem abertas. Nas palavras de Campbell, "No momento mais sombrio surge a luz."(...)"Persiga a bem-aventurança e não tenha medo, que as portas se abrirão, lá onde você nem sequer sabia que haviam portas".

Ana Lúcia Mendes Cruz Silvestre da Silva

"A soul in tension is learning to fly…"

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos Principais

$\psi(x)$ – *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.

Abreviaturas

2D – Bidimensional

AC – *Alternate Current*

ASWDR – *Adaptatively Scanned Wavelet Difference Reduction*

BPP – *Bits* por *Pixel*

DC – *Direct Current*

DFT – *Discrete Fourier Transform*, Transformada de Fourier Discreta

EZW – *Embedded Zerotree Wavelet Coding*

FIR – *Finite Impulse Response*

GFLOPS – *Giga Floating-Point Operations*

IFS – *Iterated Function System*

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*

PB – Passa Baixas

PIFS – *Partitioned Iterated Function System*

PSNR – *Peak Signal to Noise Ratio*

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

WDR – *Wavelet Difference Reduction*

Capítulo 1

Introdução

Imagens digitais exigem uma parcela cada vez maior do mundo da informação. Basta observar os avanços contínuos na tecnologia de impressoras, *scanners* e câmaras digitais; por exemplo. Nesse âmbito, o campo da compressão de imagens experimentou recentemente uma explosão de interesse devido especialmente à expansão da *Internet* e de outras aplicações multimídia.

Embora os métodos de compressão de imagens sejam de uso extensivo hoje, a demanda pelo aumento da capacidade de armazenamento e de velocidade de transmissão exige pesquisas contínuas em busca de novos métodos ou na melhoria dos já existentes. Muitas áreas da ciência, em especial a engenharia e matemática têm se dedicado a essa questão. Por esse motivo tem sido dada atenção a duas novas vertentes: tecnologia *wavelet* e tecnologia fractal. Fractais e *wavelets*, nas palavras de Wealsted [1], "providenciam dois diferentes caminhos para tais pesquisas".

Novas tecnologias, tais como fractais e *wavelets*, se tornam aliadas potenciais no estabelecimento de novos padrões de compressão de imagens. De fato, a transformada *wavelet* (TW) apresenta algoritmos rápidos, boa capacidade de concentrar a energia do sinal em poucos coeficientes, transmissão progressiva, além de não introduzir artefatos de blocagem. Essas vantajosas características permitiram a inserção da TW 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, transmissão progressiva, boa compressão e o fato de praticamente não necessitar de codificadores entrópicos. Por se tratar de uma técnica ainda pouco explorada, se apresenta como um vasto campo para a pesquisa, dado o seu alto potencial para a compressão de imagens. Um dos aspectos que encorajam a codificação fractal é que ela é completamente paralelizável. *Hardware* paralelo especializado pode, portanto, reduzir significativamente o tempo de codificação.

Apesar das vantajosas características da compressão fractal, o tempo exaustivo de processamento é a principal barreira na implementação de sistemas práticos que façam uso dessa tecnologia.

Dada essa limitação no tempo de processamento das implementações correntes, métodos fractais puros são provavelmente mais convenientes à aplicações de arquivamento, tais como enciclopédias digitais, onde uma imagem é codificada uma vez e decodificada muitas vezes. Já os métodos *wavelet* são mais convenientes para aplicações que exigem codificação rápida, tais como comunicações em tempo

real via *Internet*.

Diversas tentativas recentes tem sido feitas na tentativa de minimizar o extenso tempo de processamento fractal, abordando desde vetores de características até redes neurais. Muita pesquisa ainda se faz necessária para destravar o potencial fractal latente para a compressão de imagens. Sem dúvida nenhuma, essas pesquisas devem residir na questão de promover a aceleração da etapa de compressão dessa nova tecnologia.

1.1 – Contribuições e Organização do Trabalho

O objetivo deste trabalho é propor uma solução para a questão da velocidade computacional fractal de imagens estáticas através da aplicação da TWD e suas propriedades, sem que haja detrimento do PSNR ou da qualidade visual. Logo, este trabalho apresenta um novo Codificador Híbrido Fractal-*Wavelet*, que aplica a compressão fractal acelerada à imagens estáticas decompostas pela transformada *wavelet*, explorando a direcionalidade das subimagens-*wavelet*, constatada por Shapiro [2].

Através deste esquema, para diversas imagens e *bitrates*, obteve-se uma melhora visual corroborada pelos valores de PSNR e um aumento na velocidade de processamento fractal pura em cerca de 80% para uma mesma taxa de bits, tornando viável a exploração dessa poderosa ferramenta de compressão fractal conjunta. Nenhum artefato de blocagem (comum a procedimentos fractais puros) resultou do processo de compressão híbrido. Os detalhes da imagem e as características de transmissão progressiva *wavelet* foram também preservados. O aumento na complexidade do codificador pode ser considerado de pequeno porte.

Dessa forma, são sumarizadas aqui as principais contribuições deste trabalho:

(a) Idéia Central: exploração conjunta da codificação fractal com a direcionalidade *wavelet*. Conforme será abordado no Capítulo 12, Shapiro coloca que as subbandas-*wavelet* de um mesmo tipo representam uma mesma estrutura espacial da imagem em diferentes escalas [2]. Logo, neste trabalho, propõe-se a idéia de que os pares *range-domain* fractais de um determinado tipo de subimagem *wavelet* poderiam ser preditos pelos pares *range-domain* da subimagem-*wavelet* de menor escala do mesmo tipo. Essa direcionalidade substitui o uso de *Domain Pools* de forma bastante significativa e eficiente. A conseqüência direta é o aumento de velocidade de compressão.

(b) Implementação dos Sistemas-Base Fractal Puro e *Wavelet* Puro (para comparação). elaboração e implementação do Codificador/Decodificador Híbrido Fractal-*Wavelet* completo, contando com um esquema de controle de qualidade (que classifica os blocos para uma melhor distribuição de quantização,

permitindo que haja uma melhor codificação dos blocos conforme a necessidade) e com um mecanismo de escape (prevenindo exceções de ausência de auto-similaridade).

Os capítulos foram, então, divididos da seguinte forma:

TEORIA DE WAVELETS: Capítulos 2-7.

Capítulo 2: Transformada *Wavelet* Contínua (TWC)

Neste capítulo é feita uma apresentação dos princípios básicos da transformada *wavelet* contínua 1D e 2D, de suas características e princípios de funcionamento. A interpretação da TW via banco de filtros talvez seja o conceito mais importante deste capítulo uma vez que nele estão baseadas as aplicações dessa transformada.

Capítulo 3: Introdução aos Princípios da Transformada *Wavelet* Discreta (TWD)

Neste capítulo realiza-se um breve estudo acerca da codificação por subbanda resultando na implementação prática da TWD através dos filtros do algoritmo rápido de Mallat. Trata-se de uma preparação para, no Capítulo 4, apresentar os sinais resultantes da decomposição *wavelet*: o sinal passa-baixas (PB) e os sinais de detalhes. É sobre esses sinais (estendidos para o caso 2D e biortogonal) que será aplicada a codificação fractal do codificador híbrido proposto nesta pesquisa.

Capítulo 4: Análise Multiresolução e Projeto da TWD

Apresenta-se neste capítulo a decomposição e síntese da TWD para o caso ortogonal, bem como os sinais PB e de detalhes resultantes do processo de decomposição. Uma vez colocada a teoria ortogonal, pode-se relaxar as condições de operação, gerando a biortogonalidade, que sana as deficiências do caso ortogonal e possibilita efetivamente a aplicação da TWD na compressão de imagens com excelente desempenho. A biortogonalidade consagrou o uso da TWD na compressão de imagens e é o tema do capítulo seguinte.

Capítulo 5: Biortogonalidade

Neste capítulo são abordadas a decomposição e síntese da TWD para o caso biortogonal, que estendida para o caso bidimensional, apresenta desempenho excelente para a compressão de imagens. Esse desempenho se deve às excelentes características conjuntas dos filtros biortogonais: suporte compacto, simetria e bom compromisso entre regularidade e comprimento do filtro. Essas características serão melhor abordadas no Capítulo 7.

Capítulo 6: Análise Multiresolução Bidimensional

São detalhadas neste capítulo a decomposição e síntese da TWD para o caso bidimensional ortogonal, bem como os sinais PB (2D) e de detalhes (2D) resultantes do processo de decomposição.

Capítulo 7: Regularidade e Seleção das *Wavelets*

Este capítulo tem por objetivo apresentar as características desejáveis para a TWD na compressão de imagens, com destaque para a regularidade; culminando com a importância da biortogonalidade e nas características da TWD *spline* biortogonal 9-7, adotada por este trabalho. Também é apresentada a codificação SPIHT (*Set Partitioning in Hierarchical Trees* – Particionamento de Conjuntos em Árvores Hierárquicas), que comporá o codificador *wavelet* puro que será utilizado como o Sistema-Base 1 a ser comparado com o codificador híbrido proposto nesta pesquisa. Este capítulo encerra a abordagem da teoria de transformada *wavelet*, nas quais foram fundamentados todos os conceitos necessários à apresentação da parcela *wavelet* do Codificador Híbrido Fractal-*Wavelet* proposto neste trabalho.

TEORIA DE FRACTAIS: Capítulos: 8-11.

Capítulo 8: Bases da Compressão Fractal de Imagens

Neste breve capítulo, apresentam-se os princípios básicos da compressão fractal. São introduzidos os conceitos de transformação afim, iteração e atrator, sendo este último o mais importante. Trata-se de uma abordagem inicial com caráter ilustrativo, para num segundo momento (Capítulos 9 e 10) apresentar a abordagem matemática, da qual derivarão esses fundamentos. Esses tópicos serão importantes para compreender porque e como a compressão fractal de imagens funciona.

Capítulo 9: Espaços Métricos Completos e Contratividade

Este capítulo coloca a abordagem matemática relativa aos conceitos fractais ilustrados no Capítulo 8, com destaque para os teoremas da colagem e do mapeamento contrativo. O primeiro deles permite que, dado o atrator, possa-se determinar os coeficientes das transformações afins a serem enviados (codificação fractal). O segundo garante que o processo fractal converge para um atrator (decodificação fractal). Também é apresentado o método de geração de fractais conhecido como IFS (*Iterated Function System* – Sistema de Funções Iterativas), inspiração dos codificadores fractais hoje em uso.

Capítulo 10: Sistemas de Funções Iterativas Particionadas (PIFS)

Apresenta-se neste capítulo o sistema de codificação fractal conhecido como PIFS (*Partitioned Iterated Function Systems* - *Si*stema de Funções Iterativas Particionadas), método de codificação efetivamente usado nos codificadores fractal atuais, incluindo o codificador híbrido elaborado por este trabalho. São apresentados seus princípios de funcionamento (baseado na similaridade por partes), o particionamento da imagem em *domain-blocks* e *range-blocks*, a aplicação das transformações afins, o método básico de busca e encontro do *domain-block* equivalente e as informações a serem enviadas ao decodificador.

Capítulo 11: Codificação Fractal de Imagens Acelerada

Este capítulo apresenta os dois sistemas-base fractais para comparação com o codificador híbrido proposto neste trabalho: o Codificador de Fisher [3] e o Codificador de Cardinal [4]. Ambos também objetivam acelerar o processo de codificação fractal. O primeiro deles foi escolhido por ser um dos codificadores mais implementados na literatura científica atual. O segundo foi também escolhido por ser uma referência bastante atualizada e requisitada como um bom parâmetro de comparação pela revista científica *IEEE Transactions on Image Processing* em dezembro de 2004. A bibliografia referente a outros codificadores acelerados é também apresentada de forma a situar o leitor nas diversas possibilidades. Este capítulo encerra a abordagem da teoria de codificação fractal, na qual foram fundamentados todos os conceitos necessários à apresentação da parcela fractal do Codificador Híbrido Fractal-*Wavelet* proposto neste trabalho.

CODIFICADOR HÍBRIDO FRACTAL-WAVELET PROPOSTO
E ANÁLISE DE RESULTADOS: Capítulos: 12-14

Capítulo 12: Codificador Hìbrido *Fractal-Wavelet* Proposto e Sistemas-Base de Comparação

São detalhados neste capítulo os princípios de funcionamento do codificador proposto, cuja idéia central é a exploração conjunta da codificação fractal com a direcionalidade *wavelet*, procedimento que substitui o uso de *Domain Pools* de forma bastante significativa e eficiente e que tem como conseqüência direta o aumento de velocidade de codificação. Os Sistemas-Base de Comparação utilizados nos experimentos também são sumarizados e identificados, bem como são apresentadas as suas particularidades de implementação e as do codificador proposto. O Codificador/Decodificador Híbrido Fractal-*Wavelet* completo é apresentado, contando com um esquema de controle de qualidade e com um mecanismo de escape (prevenindo exceções de ausência de auto-similaridade).

Capítulo 13: Apresentação e Análise dos Resultados Experimentais

Neste capítulo apresentam-se os resultados das simulações e as respectivas análises realizadas durante o desenvolvimento do trabalho. Os resultados do codificador híbrido são comparados com os resultados de sistemas-base fractais "puros" e com o sistema-base *wavelet* "puro". Os resultados obtidos foram analisados quanto ao tempo de codificação, PSNR, e qualidade subjetiva para uma ampla gama de imagens conhecidas na literatura científica codificadas a vários *bitrates*.

Capítulo 14: Conclusões e Sugestões para Trabalhos Futuros

Finaliza-se este trabalho apresentando-se comentários e conclusões à luz dos resultados obtidos, analisando-se as perspectivas de melhorias e extensões do procedimento proposto que poderão ser implementadas com base nos resultados já obtidos. Espera-se, através deste trabalho e das sugestões aqui apresentadas, colaborar de alguma forma para que o potencial da compressão fractal seja expandido.

Capítulo 2

Transformada *Wavelet* Contínua (TWC)

Nos últimos anos tem crescido cada vez mais o interesse em novas técnicas, transformadas e combinações de diferentes tecnologias que resolvam os problemas da compressão de imagens, detecção de bordas, e análise de texturas.

Nesse contexto, a transformada *wavelet* (TW) foi desenvolvida na última década, fruto das contribuições de pesquisadores das mais diversas áreas, tais como matemática, física, estatística, computação gráfica e engenharia. Essas contribuições surgiram no sentido de eliminar problemas inerentes às outras transformadas, objetivando um melhor desempenho nas diversas aplicações.

A TW apresenta algoritmos de implementação rápida, boa capacidade de concentrar a energia do sinal em poucos coeficientes, além de não introduzir os efeitos de blocagem; características que a consagraram nos últimos anos como sendo uma excelente alternativa aos então métodos tradicionais de compressão de imagens. Hoje, importantes padrões de compressão de imagem e vídeo tais como o JPEG 2000, MPEG-4 e MPEG-7 fazem uso da transformada *wavelet*.

Neste capítulo serão introduzidos os princípios básicos da transformada *wavelet* contínua 1D e 2D, sua interpretação através do conceito de banco de filtros e a expansão em série da *wavelet* (SW) 1D e 2D. O estudo a seguir se restringe a funções quadraticamente integráveis 1D e 2D, uma vez que estas englobam os sinais e imagens que são de interesse.

2.1 – Introdução: *Waves* x *Wavelets*

O objetivo da análise de sinais é extrair informação relevante de um sinal. Em geral, isso é feito através de uma transformação que permita representar o sinal num domínio mais elucidativo a respeito de suas características do que o domínio do tempo.

Nos processos de transformação de sinais e imagens, sabe-se que cada coeficiente transformado é o resultado do produto interno entre a função de entrada e uma das funções-base da transformação. Assim sendo, esse valor representa, de certa forma, o grau de similaridade entre a função de entrada e determinada função-base.

Assim, se os componentes de interesse no sinal de entrada (p.e., bordas) são similares a poucas funções-base, gerariam coeficientes maiores correspondentes a essas funções-base, sendo fácilmente localizados e até mesmo removidos. Daí conclui-se o potencial em usar transformadas com funções-base que sejam similares à determinados componentes do sinal de entrada.

A transformada de Fourier (TF) é uma ferramenta clássica na análise de sinais e decompõe o sinal em funções-base ortonormais $e^{-j2\pi sx}$. A TF contínua de um sinal *f(x)* é dada por:

$$\boldsymbol{F}(\boldsymbol{s}) = \int_{-\infty}^{\infty} f(x) \cdot e^{-j2\pi sx} dx \tag{2.1}$$

Para a transformação de Fourier integral, essas funções-base senoidais tendem ao infinito em ambas as direções. Os vetores-base da transformada de fourier discreta (DFT) também não são nulos em todo o seu domínio. Logo, a TF não apresenta suporte compacto.

A rigor a TF pode representar qualquer função analítica (mesmo um sinal transiente estreito) como uma soma de senóides através de interferência destrutiva. O problema é que componentes transientes importantes do sinal (bordas naturais, por exemplo) só diferem de zero durante um curto intervalo de tempo, não se assemelhando às funções-base da TF e não sendo portanto representados de forma compacta nos coeficientes transformados (espectro em freqüência).

Para resolver esse problema, matemáticos e engenheiros têm explorado várias aproximações usando transformadas que tenham funções-base de duração limitada. Essas funções-base (chamadas *wavelets* ou ondeletas) são ondas que variam em posição e freqüência, e as transformadas baseadas nelas são chamadas de transformadas *wavelet* [5]. A Fig. 2.1 ilustra a diferença entre ondas (*waves*) e ondeletas (*wavelets*), sendo estas últimas as duas curvas inferiores.

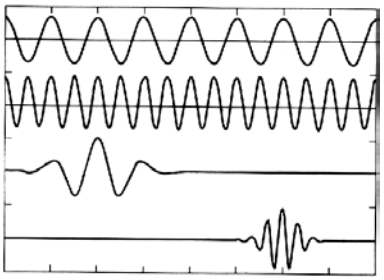


Fig. 2.1. *Waves* e *Wavelets*

A técnica conhecida por análise tempo-freqüência precedeu a TW, mas se encaixa na mesma estrutura da TW. Na análise tempo-freqüência o sinal é submetido a várias filtragens diferentes gerando, por exemplo, o gráfico da Fig. 2.2 que permite uma análise do conteúdo do sinal. A TW que será apresentada a seguir também torna possível este tipo de análise, decompondo o sinal em sua base de funções.

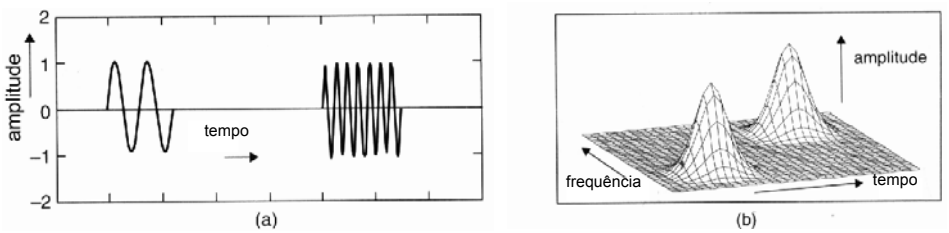


Fig. 2.2. Espaço tempo-freqüência: (a) sinal original; (b) representação

2.2 - Da Série de Fourier à TW

O espaço de funções $L^2(0,2\pi)$ é definido como a classe de funções mensuráveis (Lebesgue), definidas em $[0,2\pi]$, tais que:

$$\int_0^{2\pi} |f(x)|^2 dx < \infty \tag{2.2}$$

A representação por Série de Fourier (SF) e os coeficientes C_n de Fourier de uma função $f(x)$ qualquer são dados respectivamente por:

$$f(x) = \sum_{n=-\infty}^{\infty} C_n \cdot e^{jnx} \tag{2.3}$$

$$C_n = \frac{1}{2\pi} \int_0^{2\pi} f(x) \cdot e^{-jnx} dx \tag{2.4}$$

A série definida em (2.3) apresenta sua convergência definida em $L^2(0,2\pi)$:

$$\lim_{M,N\to\infty} \int_0^{2\pi} \left| f(x) - \sum_{n=-M}^{N} C_n \cdot e^{jnx} \right|^2 dx = 0 \tag{2.5}$$

Se a função $f(x) \in L^2(0,2\pi)$, há convergência e pode-se utilizar a representação em série (2.3). Logo, a função $f(x)$ pode ser decomposta em infinitas componentes ortonormais $g_n(x) = e^{jnx}$ [6]. Sendo $< g_m , g_n >$ o produto interno de duas componentes $g_m(x)$ e $g_n(x)$, a ortonormalidade da base $g_n(x)$ é definida por:

$$\frac{1}{2\pi} \langle g_m, g_n \rangle = \frac{1}{2\pi} \cdot \int_0^{2\pi} g_m(x) \cdot g_n^*(x) dx = \begin{cases} 0, & se\ m \neq n \\ 1, & se\ m = n \end{cases} \tag{2.6}$$

Assim, na série de Fourier, e^{jnx} é uma base ortonormal para o espaço $L^2(0,2\pi)$ gerada através de dilatações e compressões de uma única função e^{jx}. Assim, baixos valores de $|n|$ correspondem a baixas freqüências e altos valores de $|n|$ correspondem a altas freqüências.

Definindo agora o espaço $L^2(\Re)$ como sendo a classe de funções f(x) mensuráveis e

quadraticamente integráveis (sinais de energia), ou seja, funções tais que:

$$\int_{-\infty}^{\infty} |f(x)|^2 dx < \infty \tag{2.7}$$

Assim como na SF, onde o espaço $L^2(0,2\pi)$ é gerado a partir de compressões e dilatações de uma única função, na TW procuramos uma única função $\psi(x)$ capaz de gerar todo o espaço $L^2(\Re)$ através de dilatações e compressões de si mesma. Porém, ao contrário de e^{jx}, a função $\psi(x)$ tem duração finita, o que implica que para cobrir todo o espaço $L^2(\Re)$, é necessário utilizar combinações lineares dos deslocamentos de $\psi(x)$ ao longo da reta real.

A função protótipo $\psi(x)$ é chamada de *wavelet*-básica, função oscilatória usualmente centrada na origem e que cai rapidamente a zero quando $|x| \to \infty$. Assim, $\psi(x) \in L^2(\Re)$.

2.3 - Transformada *Wavelet* Contínua 1D

A TWC foi introduzida por Grossman e Morlet [7]. As funções-base $\psi_{a,b}(x)$ são comumente chamadas de *wavelets*-filhotes, versões deslocadas e dilatadas ou comprimidas da *wavelet*-mãe (ou *wavelet*-básica) $\psi(x)$ e capazes de gerar todo o espaço $L^2(\Re)$. Assim, sendo $a>0$ e b números reais:

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

Em (2.8), b é o parâmetro responsável pelo deslocamento de $\psi(x)$ ao longo do eixo x e a é o parâmetro responsável pela dilatação ou compressão. A constante $a^{-1/2}$ em (2.8) é o termo de normalização da energia da função com relação ao parâmetro a.

Na Fig. 2.3 é apresentada a *wavelet* de Morlet para diversos valores de a e b.

O fator de escala $a^{-1/2}$ em (2.8) assegura que as normas das funções-base sejam iguais, pois:

$$\left\| f\left(\frac{x-b}{a}\right) \right\| = \sqrt{\int_{-\infty}^{+\infty} \left| f\left(\frac{x-b}{a}\right) \right|^2 dx} = \sqrt{a}\, \|f(x)\| \tag{2.9}$$

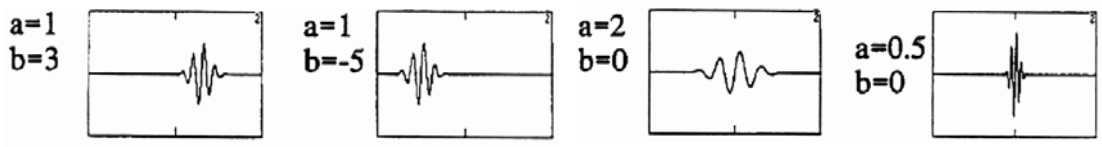


Fig. 2.3. Wavelets de Morlet deslocadas.

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 [5]:

$$W(a,b) = < f, \psi_{a,b} > \overset{\Delta}{=} \int_{-\infty}^{+\infty} f(x) \cdot \psi_{a,b}(x)\, dx \tag{2.10}$$

onde $\psi_{a,b}(x)$ é dado em (2.8).

Para se obter a TWC inversa, i.e., reconstruir $f(x)$ a partir de seus coeficientes *wavelet*, é necessário que o seguinte critério de admissibilidade seja satisfeito [8] [9]:

$$C_{\psi} = \int_{-\infty}^{+\infty} \frac{|\Psi(s)|^2}{|s|}\, ds < \infty \tag{2.11}$$

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:

$$\Psi(0) = 0 = \int_{-\infty}^{+\infty} \psi(x)\, dx = 0 \tag{2.12}$$

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

Se a constante C_{ψ} na equação (2.11) for finita, então existe a TWC inversa definida por:

$$f(x) = \frac{1}{C_{\Psi}} \int_0^{\infty} \int_{-\infty}^{+\infty} W(a,b)\, \psi_{a,b}(x)\, db \frac{da}{a^2} \tag{2.13}$$

Como a *wavelet*-básica tem média nula, todos os escalonamentos e translações de (2.8) também têm média nula.

2.4 – Interpretação da TWC por Banco de Filtros

A TWC possui janela de comprimento variável, ou seja, resoluções no tempo e freqüência variáveis e também pode ser interpretada através do conceito de banco de filtros [5]. Definindo a *wavelet* escalonada de a e normalizada por $a^{-1/2}$:

$$\psi_a(x) = \frac{1}{\sqrt{a}} \psi\left(\frac{x}{a}\right) \tag{2.14}$$

Definindo também seu complexo conjugado refletido:

$$\tilde{\psi}_a(x) = \psi_a^*(-x) = \frac{1}{\sqrt{a}} \psi^*\left(\frac{-x}{a}\right) \tag{2.15}$$

Pode-se reescrever (2.10) e (2.13), respectivamente como:

$$W(a,b) = \int_{-\infty}^{+\infty} f(x) \psi_a(b-x)\, dx = f * \tilde{\psi}_a \tag{2.16}$$

$$f(x)=\frac{1}{C_\Psi}\int_0^\infty \int_{-\infty}^{+\infty}[f*\tilde{\psi}_a](b)\psi_a(b-x)\,db\frac{da}{a^2}=\frac{1}{C_\Psi}\int_0^\infty [f*\tilde{\psi}_a*\psi_a](x)\frac{da}{a^2} \qquad (2.17)$$

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 (2.14). A TWC pode ser interpretada como um banco de filtros lineares atuando sobre *f(x)*, conforme ilustrado na Fig. 2.4. Todas as saídas dos filtros juntas compõem a TWC.

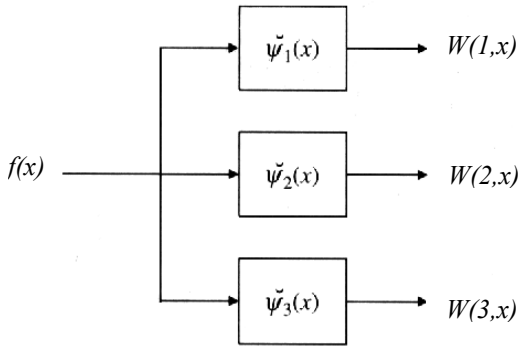


Fig. 2.4. Interpretação da TWC por banco de Filtros

Deve-se notar que a TW não envolve explicitamente o conceito de freqüência. Apesar da relação implícita existente entre escala e freqüência, esses conceitos são diferentes, modificando a forma de interpretação dessas duas transformadas. Contudo, serão feitas analogias entre ambas a título de esclarecimentos do mecanismo de funcionamento da TW que, como será visto à frente, permite uma visão simultânea dos resultados da transformação para diversos valores de escala.

Deve-se observar que:

- Há para o esquema da Fig. 2.4 uma única *wavelet*-básica.
- $\tilde{\psi}_1(x),\ \tilde{\psi}_2(x),\tilde{\psi}_3(x),\tilde{\psi}_4(x)$ 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 uma é ainda um conjunto de funções-base de mesma "freqüência".
- Cada saída da Fig. 2.4 é, portanto, uma convolução de *f(x)* com uma *wavelet*-básica escalonada. Logo, para *a* fixo, cada *W(a,b)* é um conjunto de coeficientes de mesma freqüência, correspondendo a uma linha do gráfico da Fig. 2.2.b.
- Como em (2.15), se $\psi(x)$ for real e par (que é usualmente o caso) a reflexão e o conjugado não têm efeito, logo $\tilde{\psi}_a(x)=\psi_a(x)$.

De (2.17) percebe-se que as saídas dos filtros, cada qual filtrada novamente por $\psi_a(x)$ e corretamente escalonada, se combinam para reconstruir $f(x)$.

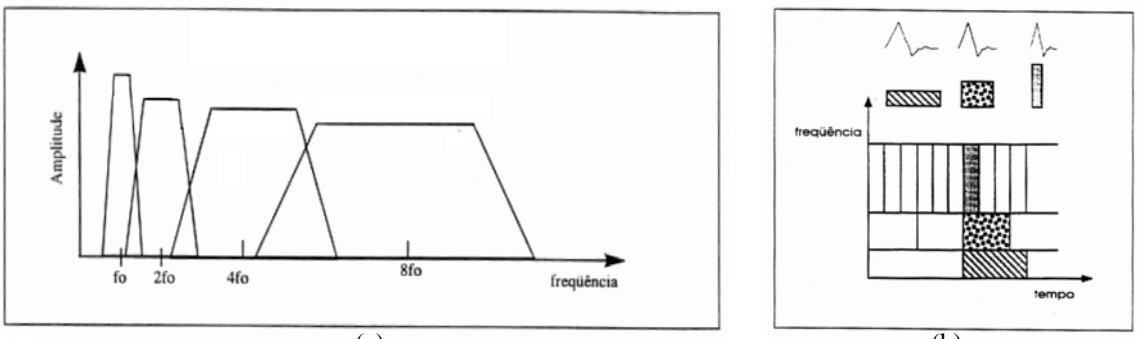
Lembrando que:

$$TF\{f(ax)\} = \frac{1}{|a|} F\left(\frac{s}{a}\right) \tag{2.18}$$

tem-se:

$$\Psi_a(s) = TF\{\psi_a(x)\} = \sqrt{a}\,\Psi(as) \tag{2.19}$$

Na TWC os filtros passa-faixas apresentam simultaneamente larguras diferentes definidas pelo valor de a. Para valores altos de a, a *wavelet* definida em (2.14) sofre dilatação no tempo e conseqüente 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. 2.5.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. 2.5. TWC: (a) Bandas de freqüências das janelas; (b) Resolução no plano tempo-freqüência [6]

A largura da *wavelet* está intimamente ligada à resolução do sinal. *Wavelets* mais largas implicam na utilização de um número maior de componentes do sinal original na formação de cada coeficiente transformado durante o processo de convolução (menor resolução no tempo e maior resolução em freqüência). Já *wavelets* mais estreitas implicam na utilização de um número menor na formação de cada coeficiente transformado (maior resolução no tempo e menor em freqüência).

A resolução da TWC, ilustrada na Fig. 2.5.b, permite a análise de um sinal sob diversas resoluções diferentes simultaneamente permitindo, por exemplo, o processamento por camadas de detalhamento do sinal [6]. Esta é a grande vantagem da TWD sobre as demais transformadas.

2.5 – TWC Bidimensional

A TWC, *W(a,b)*, de uma função *f(x)* é uma função de duas variáveis. Assim, a TWC é dita ser "sobrecompleta" pois representa um aumento considerável no volume de informação e no volume necessário para armazenamento dos dados. 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, a TWC-2D de *f(x,y)* é dada por [5]:

$$W(a;b_x,b_y)=\left\langle f(x,y),\psi_{a;b_x,b_y}(x,y)\right\rangle=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}f(x,y)\,\psi_{a;\,b_x,b_y}(x,y)\,dxdy \tag{2.20}$$

onde b_x e b_y especificam a translação em duas dimensões. $\psi_{a;b_x,b_y}(x,y)$ são as *wavelets*-filhotes bidimensionais geradas a partir da *wavelet*-básica bidimensional $\psi(x,y)$ por:

$$\psi_{a;b_x,b_y}(x,y)=\frac{1}{a}\psi\left(\frac{x-b_x}{a},\frac{y-b_y}{a}\right) \tag{2.21}$$

Para obter a TWC 2D inversa é necessário que a condição de admissibilidade seja satisfeita [6]. Assim, seja $\Psi(s_x,s_y)$ a transformada de Fourier 2D de $\psi(x,y)$, essa condição é dada por:

$$C_\psi=\int_0^\infty\int_0^{2\pi}\frac{\left|\Psi(r\cos\theta,rsin\theta)\right|^2}{r}dr\,d\theta<\infty \tag{2.22}$$

Caso $\psi(x,y)$ seja esfericamente simétrica, sua transformada de Fourier também é esfericamente simétrica. Logo, a condição de admissibilidade pode ser simplificada e escrita da seguinte forma [9]:

$$C_\psi=\int_0^\infty\frac{\left|\Psi(a^{-1}s_x,a^{-1}s_y)\right|^2}{a}da<\infty \qquad ,\forall(s_x,s_y)\in R^2 \tag{2.23}$$

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

$$f(x,y)=\frac{1}{C_\Psi}\int_0^\infty\int_{-\infty}^{\infty}\int_{-\infty}^{+\infty}W(a;b_x,b_y)\,\psi_{a;b_x,b_y}(x,y)\,db_x db_y\frac{da}{a^3} \tag{2.24}$$

Generalizações análogas são válidas para funções de mais que duas variáveis.

Deve-se notar que, em virtude da redundância (por ser sobrecompleta), por si só o valor potencial da TWC não reside em representação compacta, mas sim na decomposição e análise. Pode-se, por exemplo, projetar uma TWC de forma a permitir o acesso aos componentes da imagem isoladamente. Uma das aplicações poderia ser, por exemplo, permitir realizar um determinado processamento somente sobre os componentes de alta freqüência.

2.6 – Expansão em Série da *Wavelet* (SW)

A Expansão em série *wavelet* (SW) é obtida através da discretização dos parâmetros *a* (escala) e *b* (deslocamento), mantendo a variável *x* (tempo) contínua. Assim, o resultado da SW é um conjunto enumerável de coeficientes, que correspondem a pontos na grade bidimensional do domínio deslocamento-escala [6]. O parâmetro *a* sofre discretização exponencial, enquanto o parâmetro *b* sofre uma discretização proporcional a *a*. Respectivamente, sendo *j* e *n* inteiros:

$$a = a_0^j \tag{2.25}$$

$$b = k \cdot b_0 \cdot a_0^j = n \cdot a_0^j \tag{2.26}$$

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

Para esclarecer o fato do parâmetro *b* sofrer uma discretização proporcional ao parâmetro *a*, deve-se ter em mente a necessidade de evitar a geração de redundância. Para valores de *a* maiores, é maior o número de componentes do sinal original para formar cada coeficiente transformado. Conseqüentemente, para evitar o excesso de redundância durante a convolução com *f(x)*, os passos de deslocamento da *wavelet* devem ser maiores. Similarmente, para valores pequenos de *a*, são necessários passos de deslocamento menores.

Substituindo (2.25), (2.26) e (2.15) em (2.16), tem-se a equação da SW de um sinal *f(x)* [8]:

$$W(j,n) = a_0^{-j/2} \int_{-\infty}^{+\infty} f(x) \cdot \psi\left(a_0^{-j} \cdot x - n\right) dx \tag{2.27}$$

Por questão de notação, daqui em diante *W(j,n)* será referenciado simplesmente como $W_{j,n}$. 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.

2.6.a – *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. Para esse valor de a_0 são formadas funções-base a partir da *wavelet*-básica através de escalonamentos binários (de 2^j) e deslocamentos diádicos (de $k/2^j$). São as chamadas *wavelets* diádicas, que são definidas por [5]:

$$\psi_{j,n}(x) = 2^{j/2} \cdot \psi\left(2^j \cdot x - n\right) \qquad j,n \in \mathrm{Z} \tag{2.28}$$

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 $W_{j,n}$.

A Fig. 2.6 mostra a grade de amostragem no plano deslocamento-escala para *wavelets* diádicas. Cada nó é um coeficiente-*wavelet* associado a uma *wavelet* $\psi_{j,n}(x)$.

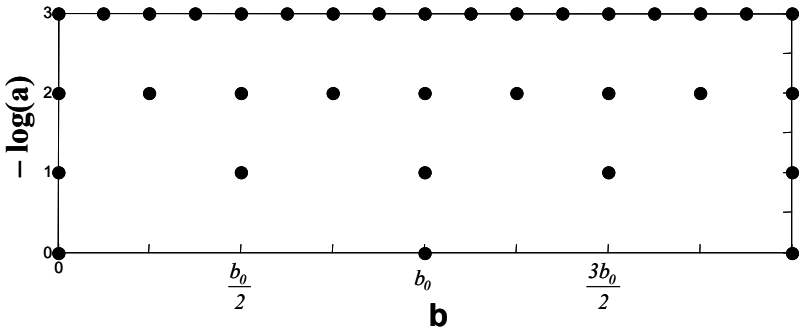


Fig. 2.6. Grade de amostragem no plano deslocamento-escala para wavelets diádicas ($a_0=1/2$)

Sabe-se que é possível reconstruir uma função de energia finita a partir dos coeficientes da TWC. No caso da SW há o procedimento análogo, porém são necessários alguns cuidados. Por exemplo, imagine duas *wavelets* com escalas a_1 e a_2 respectivamente, sendo $a_1 > a_2$. A Fig. 2.7 mostra um esboço da TF de ambas.

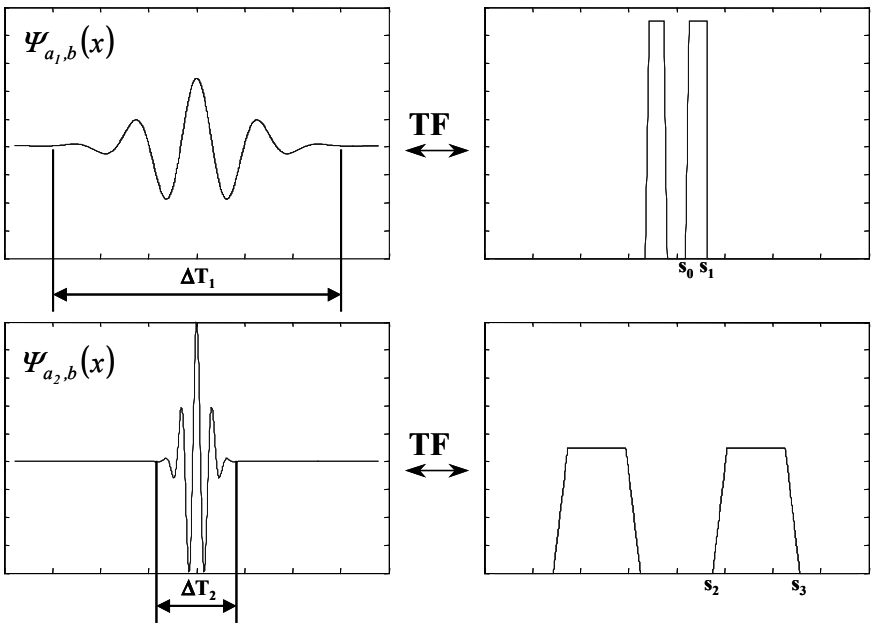


Fig. 2.7. Comparação entre as TF de duas wavelets com diferentes escalas *a*, sendo $a_1 > a_2$ e $\Delta T_1 > \Delta T_2$.

Variando *b*, pode-se interpretar ambas as *wavelets* como sendo uma seqüência horizontal de pontos da Fig. 2.6. Assim, pode-se notar que uma grade muito esparsa verticalmente indicaria uma diferença muito grande entre as escalas consecutivas de duas *wavelets*, o que poderia representar $s_2 >> s_1$ na Fig. 2.6 dependendo da escolha da *wavelet*-básica. Este fato pode vir a provocar perda de informação durante o processo de filtragem e a partir daí a equação de reconstrução pode não ser mais válida.

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.

Assim, uma função $f(x) \in L^2(R)$ pode ser reconstruída a partir dos coeficientes-*wavelet* $W_{j,n}$ da seguinte forma [8]:

$$f(x) = \sum_{j=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} W_{j,n} \cdot \psi_{j,n}(x) \tag{2.29}$$

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

$$\left\langle \psi_{j,k} , \psi_{l,m} \right\rangle = \delta_{j,l} \cdot \delta_{k,m} = \begin{cases} 1, & se \;\; j = l, k = m \\ 0, & cc \end{cases} \tag{2.30}$$

sendo *l, m* inteiros e $\delta_{j,n}$ sendo a função delta de Kronecker. Os coeficientes da série podem ser obtidos por:

$$W_{j,n} = \left\langle f, \psi_{j,n} \right\rangle \tag{2.31}$$

A série em (2.29) converge em $L^2(R)$:

$$\lim_{M_1,N_1,M_2,N_2 \to \infty} \left\| f(x) - \sum_{j=-M_2}^{N_2} \sum_{j=-M_1}^{N_1} W_{j,n} \cdot \psi_{j,n} \right\| = 0 \tag{2.32}$$

Assim, para as *wavelets* diádicas, os coeficientes da SW são dados por [8]:

$$W_{j,n} = \left\langle f, \psi_{j,n} \right\rangle = 2^{j/2} \int_{-\infty}^{\infty} f(x) \cdot \psi\left(2^j x - n\right) dx \tag{2.33}$$

Em (2.29) 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.

2.6.b – SW bidimensional

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

$$W_{j\,;\,n_x,n_y} = a^{-j} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y) \cdot \psi\left(a_0^{-j} x - n_x \, , a_0^{-j} y - n_y\right) dxdy \tag{2.34}$$

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

$$\psi_{j\,;\,n_x,n_y}(x,y) = 2^j \psi\left(2^j x - n_x \, , 2^j y - n_y \right) \tag{2.35}$$

2.7 – Comentários

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, suas características e princípios de funcionamento. A interpretação da TW via banco de filtros é fundamental e talvez seja o conceito mais importante, na medida em que as aplicações dessa transformada residem nesse conceito. Também foram abordadas a série da *wavelet* uni e bidimensional como uma introdução à tranformada *wavelet* discreta, tema do capítulo seguinte e cujo caso bidimensional é parte central deste trabalho.

Capítulo 3

Introdução aos Princípios da Transformada *Wavelet* Discreta

A transformada *wavelet* discreta, um dos núcleos do codificador proposto neste trabalho, apresenta um ótimo desempenho na compressão, processamento e análise de imagens. Antes de introduzir a TWD propriamente dita, será abordada uma técnica que levou ao desenvolvimento de sua implementação: a codificação por subbanda [5]. Por fim, neste capítulo é apresentado o algoritmo rápido de Mallat [10] que é efetivamente utilizado para a implementação da TWD.

3.1 – Codificação por Subbanda

A codificação por subbanda é uma técnica tempo-freqüência originalmente desenvolvida para possibilitar a codificação compacta de sinais de áudio digitais. Essa técnica 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 [5] [11].

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

$$\boldsymbol{F}(s) = 0 \quad para \quad |s| \geq s_N \tag{3.1}$$

Na Fig. 3.1.a, pode-se amostrar esse sinal no tempo à freqüência de Nyquist, $S_{Nyquist}$, dada por:

$$S_{Nyquist} = \frac{1}{\Delta x} = 2 \cdot s_N \tag{3.2}$$

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 a dois filtros que gerarão dois sinais de subbanda: um sinal meia-banda baixa e um sinal meia-banda alta.

Sinal Meia-Banda Baixa

Inicialmente o espectro do sinal $f(i\Delta x)$, que é periódico de período $2s_N$, é submetido ao filtro passa-baixas meia-banda $h(i\Delta x)$, assim chamado por deixar passar somente a banda $[-s_N/2, s_N$

/2] , ou seja, a metade da banda de freqüências do sinal. Como resultado é produzido $y_0(i.\Delta x)$, uma versão de baixa resolução ("borrada") de $f(i\Delta x)$, como pode ser observado na Fig. 3.1.c.

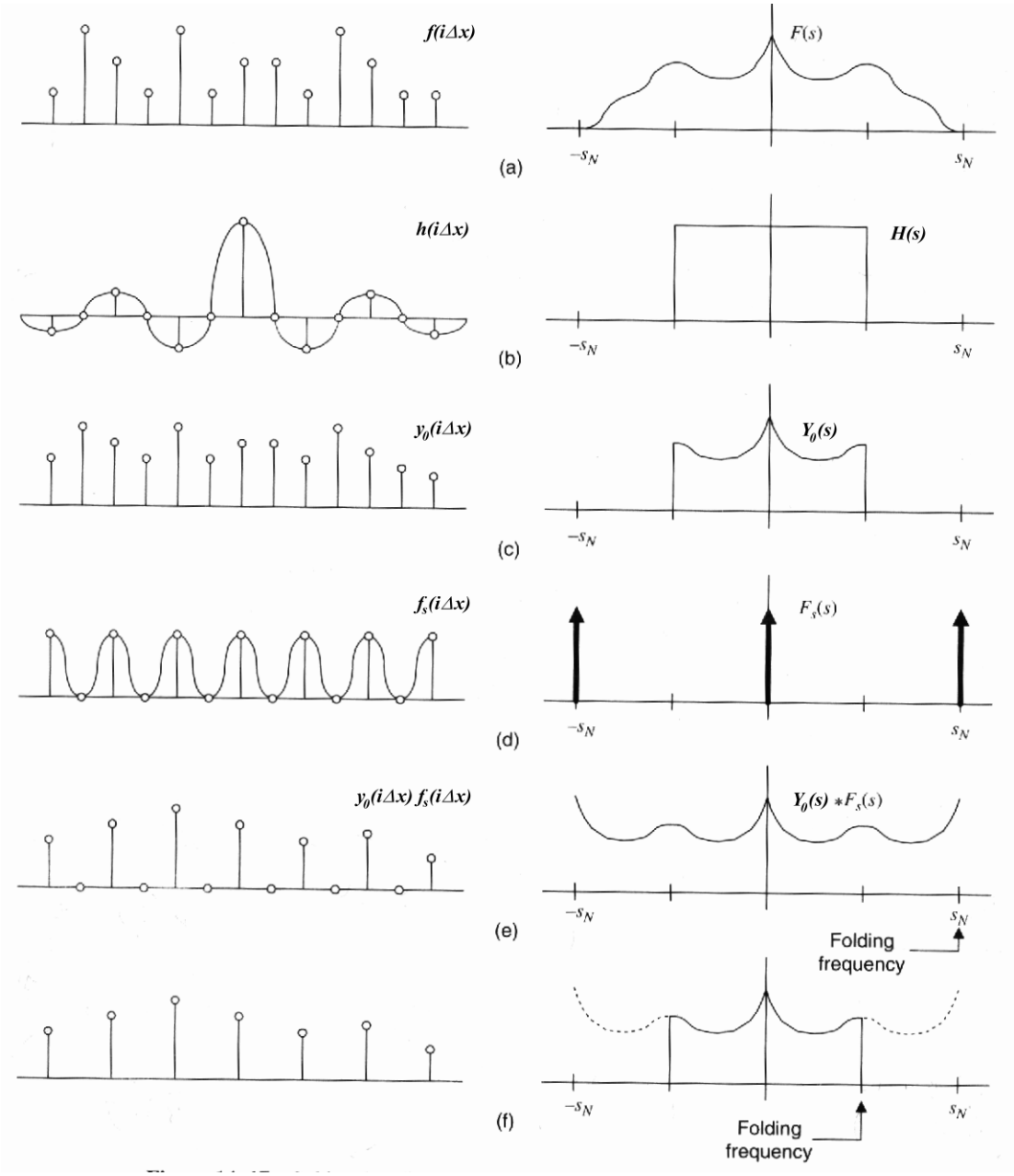


Fig. 3.1. Meia-banda baixa da codificação por subbanda: (a) sinal amostrado e seu espectro; (b) filtro passa-baixas meia banda ideal; (c) sinal filtrado; (d) função de subamostragem; (e) amostras ímpares substituídas por zeros; (f) amostras ímpares descartadas.

A resposta ao impulso e a função de transferência do filtro são respectivamente:

$$h(x) = sinc\left(\pi \frac{x}{2\Delta x}\right) \tag{3.3}$$

$$H(s) = \prod\left(\frac{s}{s_N}\right) \text{ , onde:} \tag{3.4}$$

$$\prod(x) = \begin{cases} 1 & se\ |x| < 1/2 \\ 1/2 & se\ |x| = 1/2 \\ 0 & se\ |x| > 1/2 \end{cases} \tag{3.5}$$

Com a aplicação da filtragem, o sinal $y_0(i.\Delta x)$ fica limitado ao intervalo $[-s_N/2,\ s_N/2]$, podendo agora sofrer o processo chamado de subamostragem ou decimação. No caso, $y_0(i.\Delta x)$ pode ser amostrado com um novo espaçamento de até $2\Delta x$, ou seja pode ser representado com a metade das amostras sem que haja perda de informação.

Essa subamostragem pode ser modelada através da aplicação de uma função de subamostragem que zere as amostras ímpares para permitir o descarte das mesmas. A função pode ser vista na Fig. 3.1.d e pode ser escrita como:

$$f_s(i.\Delta x) = \frac{1}{2}\left[1 + \cos(2\pi\, s_N\, i\Delta x)\right] \tag{3.6}$$

cujo espectro é dado por:

$$F_s(s) = \frac{1}{2}\left[\delta(s) + \delta(s - s_N) + \delta(s + s_N)\right] \tag{3.7}$$

A aplicação da subamostragem reduz o período de $y_0(i.\Delta x)$ de $2s_N$ para s_N, como pode ser observado na Fig. 3.1.e em virtude da convolução de $Y_0(s)$ com $F_s(s)$. Assim:

$$F_s(s) * Y_0(s) = \frac{1}{2}Y_0(s) + \frac{1}{2}Y_0(s + s_N) + \frac{1}{2}Y_0(s - s_N) \tag{3.8}$$

Após essa convolução, pode-se então descartar as amostras ímpares sem que haja nenhuma perda de informação, como pode ser observado na Fig. 3.1.f. Pode-se então representar $y_0(i.\Delta x)$ com a metade do número de amostras.

Uma forma simples de recuperar $y_0(i.\Delta x)$ a partir do sinal da Fig. 3.1.f é primeiramente superamostrar o sinal da Fig. 3.1.f inserindo-se amostras ímpares nulas, formando o sinal da Fig.3.1.e. A seguir, aplicando-se um filtro $2h(i\Delta x)$, pode-se reconstruir o espectro e daí o sinal $y_0(i.\Delta x)$:

$$F_s(s) * Y_0(s) \times H(s) = \left[\frac{1}{2}Y_0(s) + \frac{1}{2}Y_0(s + s_N) + \frac{1}{2}Y_0(s - s_N)\right] \times \prod\left(\frac{s}{s_N}\right) = \frac{1}{2}Y_0(s) \tag{3.9}$$

Sinal meia-banda alta

Para construir o sinal meia-banda alta, o sinal original *f(iΔx)* (Fig. 3.2.a) é submetido ao filtro passa-faixas meia-banda *g(iΔx)* mostrado na Fig. 3.2.b. Como resultado é produzido $y_1(i.\Delta x)$, uma versão que contém basicamente os detalhes de *f(iΔx)*, como pode ser observado na Fig. 3.2.c.

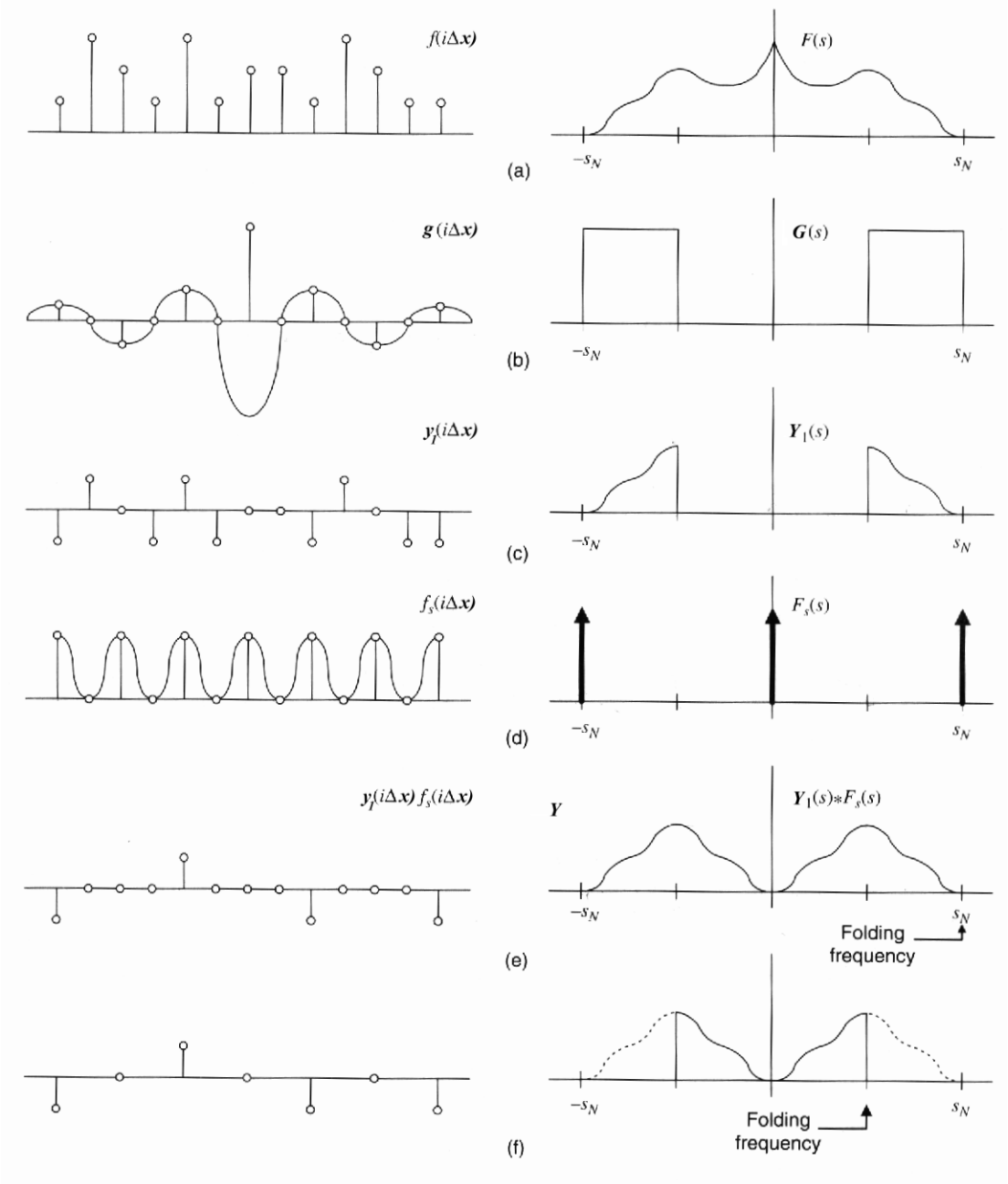


Fig. 3.2. Meia-banda alta da codificação por subbanda: (a) sinal amostrado e seu espectro; (b) filtro passa-alta meia banda ideal; (c) sinal filtrado; (d) função de subamostragem; (e) amostras ímpares substituídas por zeros; (f) amostras ímpares descartadas.

A resposta ao impulso e a função de transferência do filtro são respectivamente:

$$g(x) = \delta(x) - sinc\left(\pi \frac{x}{2\Delta x}\right) \tag{3.10}$$

$$G(s) = 1 - \prod\left(\frac{s}{s_N}\right) \tag{3.11}$$

onde $\Pi(x)$ é dado em (3.5).

A seguir, da mesma forma que $y_0(i.\Delta x)$, o sinal $y_1(i.\Delta x)$ passará pelo processo de subamostragem. Nesse processo, $y_1(i.\Delta x)$ também é amostrado com um espaçamento de $2\Delta x$, ou seja, também é representado com a metade das amostras. A função de subamostragem é a mesma que foi aplicada ao sinal meia-banda baixa, podendo ser novamente observada na Fig. 3.2.d. A subamostragem no tempo provoca a seguinte convolução em freqüência:

$$F_s(s) * Y_1(s) = \frac{1}{2} Y_1(s) + \frac{1}{2} Y_1(s + s_N) + \frac{1}{2} Y_1(s - s_N) \tag{3.12}$$

Novamente a subamostragem reduz o período de $2s_N$ para s_N. No espectro mostrado na Fig. 3.2.e pode-se observar que para o sinal meia-banda alta, a subamostragem preenche o intervalo $[-s_N/2, s_N/2]$ com uma réplica do espectro de $y_1(i.\Delta x)$. Contudo, como esse intervalo estava vago, a subamostragem que foi aplicada não produz superposição de espectros (*aliasing*), não provocando distorções no sinal, como pode ser observado na Fig. 3.3.

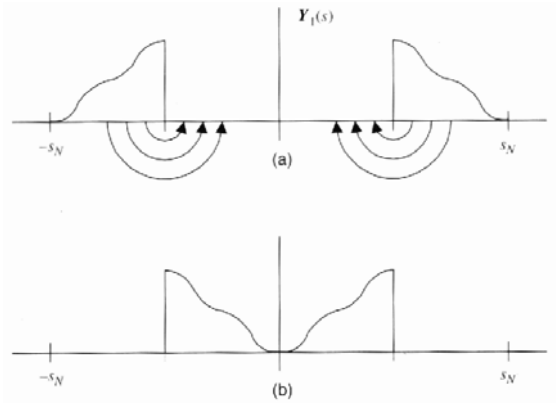


Fig. 3.3. *Aliasing* da meia banda alta: (a) formação;
(b) resultado da sobreposição.

Após essa convolução, pode-se então descartar as amostras ímpares sem que haja perda de informação, como pode ser observado na Fig. 3.2.f. Pode-se então também representar $y_1(i.\Delta x)$ com a metade do número de amostras.

O sinal $y_1(i.\Delta x)$ pode ser recuperado da mesma forma que $y_0(i.\Delta x)$, ou seja, a partir do sinal da Fig. 3.2.f. Assim, basta primeiramente superamostrar o sinal da Fig. 3.2.f inserindo amostras ímpares nulas e em seguida aplicar um filtro $2g(i\Delta x)$:

$$F_s(s) * Y_1(s) \times G(s) = \left[\frac{1}{2}Y_1(s) + \frac{1}{2}Y_1(s+s_N) + \frac{1}{2}Y_1(s-s_N)\right] \times \left[1 - \prod\left(\frac{s}{s_N}\right)\right] = \frac{1}{2}Y_1(s) \qquad (3.13)$$

Codificação e decodificação por subbanda

Sabendo que $y_1(i.\Delta x)$ contém exatamente a informação de alta freqüência que foi eliminada de $f(i.\Delta x)$ durante a geração de $y_0(i.\Delta x)$, pode-se dizer que $y_1(i.\Delta x)$ e $y_0(i.\Delta x)$ contém juntos toda a informação presente no sinal original $f(i.\Delta x)$ [5]. Assim, tem-se:

$$f(i \cdot \Delta x) = y_0(i \cdot \Delta x) + y_1(i \cdot \Delta x) = f(i \cdot \Delta x) * h(i \cdot \Delta x) + f(i \cdot \Delta x) * g(i \cdot \Delta x) \qquad (3.14)$$

pois

$$H(s) + G(s) = 1 \qquad (3.15)$$

Logo, o sinal original $f(i.\Delta x)$ de N pontos pode ser codificado sem redundância em dois sinais de $N/2$ pontos: um sinal meia-banda baixa e um sinal meia-banda alta. Como foi colocado que $y_0(i.\Delta x)$ e $y_1(i.\Delta x)$ podem ser completamente recuperados a partir dos sinais codificados em subbanda, $f(i.\Delta x)$ pode ser reconstruído sem erros no decodificador.

Formulando matematicamente o processo completo, tem-se que a codificação por subbanda em dois canais requer somente a filtragem de $f(i.\Delta x)$ com $h(i.\Delta x)$ e $g(i.\Delta x)$, seguidas pela subamostragem de cada saída. Logo,

$$y_0(2i \cdot \Delta x) = \sum_k f(k \cdot \Delta x) \cdot h((-k + 2i)\Delta x) \qquad (3.16)$$

$$y_1(2i \cdot \Delta x) = \sum_k f(k \cdot \Delta x) \cdot g((-k + 2i)\Delta x) \qquad (3.17)$$

Para a reconstrução do sinal original basta superamostrar os dois sinais codificados em subbanda, interpolá-los com $2h(i.\Delta x)$e $2g(i.\Delta x)$, e em seguida, somá-los, ou seja:

$$f(i \cdot \Delta x) = 2\sum_k \left[y_0(2k \cdot \Delta x) \cdot h((-i + 2k)\Delta x) + y_1(2k \cdot \Delta x) \cdot g((-i + 2k)\Delta x)\right] \qquad (3.18)$$

O processo completo encontra-se esquematizado na Fig. 3.4. Deve-se notar que em $s=s_N/2$ há um pequeno problema, uma vez que a codificação e decodificação filtram $f(i.\Delta x)$ neste ponto duas vezes, uma com $h(i.\Delta x)$ e outra com $g(i.\Delta x)$ e em virtude de $H(s_N/2)=1/2$ e $G(s_N/2)=1/2$. Este problema pode ser evitado fazendo $\Pi(\pm 1/2)=(1/2)^{1/2}$ em (3.5).

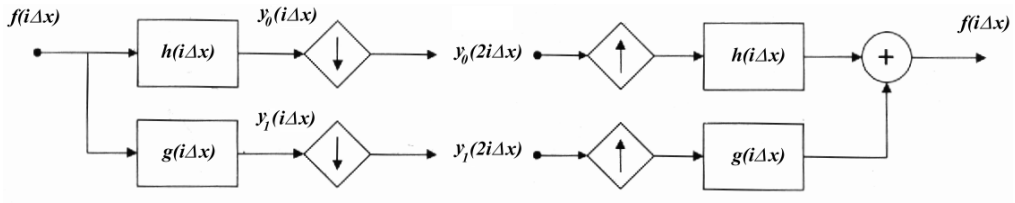


Fig. 3.4. Codificação por subbanda de duas bandas e reconstrução.

3.2 – Introdução ao Algoritmo Rápido para TWD

Mallat [10] definiu um algoritmo iterativo para a TWD que é mais eficiente que calcular o conjunto completo de produtos internos. No algoritmo, após aplicar a codificação por subbanda em dois canais ao sinal $f(i.\Delta x)$ descrita na Seção 3.1, o sinal meia-banda baixa $y_0(i.\Delta x)$ é novamente sujeito 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 é iterativo, a cada passo retendo o sinal meia-banda alta e processando o sinal meia-banda baixa, até que seja obtido um único coeficiente conforme esquematizado na Fig. 3.5. Os coeficientes transformados são então o coeficiente banda-baixa e o conjunto de coeficientes meia-banda alta codificados; que totalizarão N coeficientes.

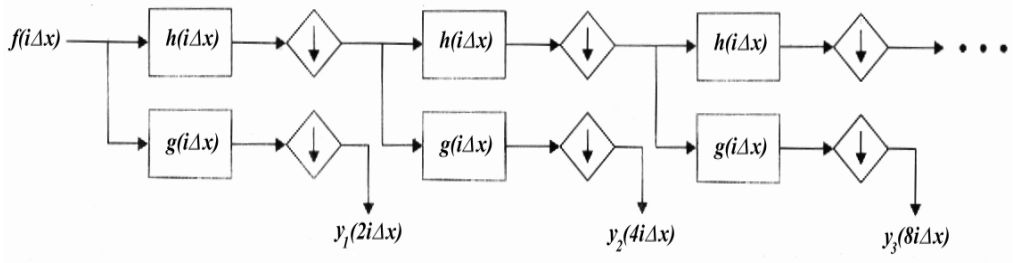


Fig. 3.5. Algoritmo de Mallat (TWD)

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 algoritmo apresentado é também chamado de FWT (*Fast Wavelet Transform* - Transformada *Wavelet* Rápida) ou *Mallat's Herringbone Algorithm*. O processo de reconstrução de

f(i.Δx) encontra-se esquematizado na Fig. 3.6. O objetivo dessa subseção é colocar o princípio de funcionamento da implementação da TWD para detalhá-la nas próximas seções.

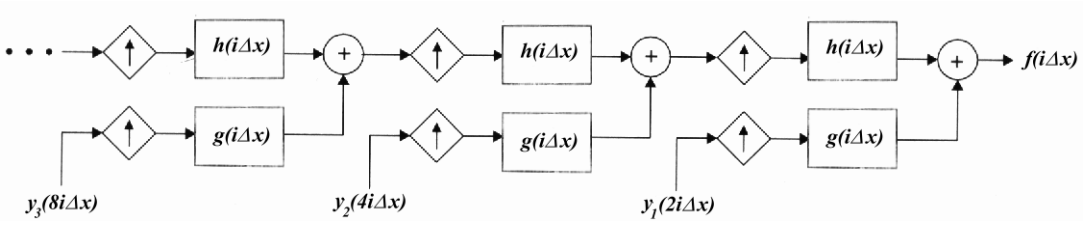


Fig. 3.6. Algoritmo inverso de Mallat (TWD)

3.3 – Comentários

Abordada a TWD e a introdução à implementação prática através do algoritmo rápido de Mallat, pode-se no Capítulo 4 apresentar essa implementação de forma completa através da análise multiresolução. A análise multiresolução permite determinar os filtros *h(i.Δx)* e *g(i.Δx)* de Mallat a partir das funções-base da TWD. São por fim gerados os sinais efetivos resultantes da decomposição *wavelet*: o sinal passa-baixas e os sinais de detalhes.

Capítulo 4

Análise Multiresolução e Projeto da TWD

A análise multiresolução é uma técnica que permite analisar o sinal através da decomposição de seu conteúdo em um conjunto de detalhes em diferentes resoluções. A resolução de um sinal $f(x) \in L^2(R)$ definido no intervalo $0< x <x_0$ é a quantidade de pontos para os quais ele é definido [6][10]. Genericamente, um sinal tem resolução r_j quando é definido para r_j pontos do intervalo. Uma função $f(x)$ contínua possui resolução infinita, uma vez que é conhecida para todos os pontos do intervalo.

Ao alterar a resolução do sinal $f(x)$ original, o sinal resultante dessa mudança é chamado de uma *aproximação* de $f(x)$. Dada uma seqüência crescente de resoluções $\{r_j , j\in Z\}$ adotadas para o sinal, os detalhes com uma dada resolução r_j do sinal são definidos como a diferença de informação entre a aproximação com resolução r_j e a aproximação com resolução r_{j-1} , mais baixa [10].

O objetivo desta seção é fornecer as bases completas da implementação da TWD através da análise multiresolução, gerando os sinais de aproximação e de detalhes após a transformação *wavelet*. Primeiramente, neste capítulo é abordado o caso ortogonal.

4.1 – Aproximação por Multiresolução em $L^2(R)$: propriedades

Seja o sinal original $f(x)\in L^2(R)$. O operador linear A_{2^j} é responsável pela projeção de $f(x)$, gerando sua aproximação com resolução 2^j . Como 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 $]0,x_0[$. A aproximação com resolução 2^j é denotada por $A_{2^j} \cdot f(x)$ e o operador apresenta as seguintes propriedades [10]:

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 $V_{2^j} \subset L^2(R)$. O subespaço vetorial V_{2^j} pode ser interpretado como o conjunto de todas as possíveis aproximações de todas as funções

$L^2(R)$ 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.

Propriedade 2 Há várias aproximações da função *f(x)* no nível de resolução 2^j. Se A_{2^j} for encarado como o pente de Dirac, basta pensar que as diferentes aproximações seriam geradas por diferentes deslocamentos desse pente. Dentre todas as aproximações de *f(x)* no nível de resolução 2^j, $A_{2^j} \cdot f(x)$ será daqui em diante designada como sendo a função que mais se aproxima de *f(x)*, ou seja:

$$\forall y(x) \in V_{2^j} \ , \ |y(x) - f(x)| \geq |A_{2^j} \cdot f(x) - f(x)| \tag{4.1}$$

Como $A_{2^j} \cdot f(x)$ é a função que mais se aproxima de *f(x)*, o operador A_{2^j} calcula então 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 necessária para calcular a sua aproximação no nível de resolução 2^j. Uma vez que A_{2^j} é um operador de projeção em V_{2^j}, tem-se que:

$$\forall j \in Z \ , \ V_{2^j} \subset V_{2^{j+1}} \tag{4.2}$$

Através da aplicação de recursão em (4.2), obtém-se a seguinte hierarquia de subespaços:

$$\ldots \subset V_{2^{-1}} \subset V_{2^0} \subset V_{2^1} \subset V_{2^2} \subset \ldots \tag{4.3}$$

Propriedade 4 A operação aproximação não depende do nível de resolução. Os subespaços das aproximações podem ser obtidos a partir de outros subespaços, através da dilatação ou compressão das aproximações:

$$\forall j \in Z \ , \ A_{2^j} \cdot f(x) \in V_{2^j} \Rightarrow A_{2^j} \cdot f\left(\frac{x}{2}\right) \in V_{2^{j+1}} \tag{4.4}$$

Propriedade 5 Caso o intervalo $0 \leq x \leq x_0$ no qual a *f(x)* é definida seja normalizado 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, invariante com o deslocamento.

Propriedade 6 Ao ser feita uma aproximação, alguma informação da função original *f(x)* é perdida, e essa perda é inversamente proporcional à resolução. Assim:

$$\lim_{j \to +\infty} V_{2^j} = \bigcup_{j=-\infty}^{+\infty} V_{2^j} = L^2(R) \tag{4.5}$$

$$\lim_{j \to -\infty} V_{2^j} = \bigcap_{j=-\infty}^{+\infty} V_{2^j} = \{0\} \tag{4.6}$$

De (4.5) e (4.6), percebe-se que à medida que a resolução aumenta tendendo a +∞, a aproximação converge para o sinal original. Similarmente, à medida que a resolução diminui a aproximação passa a conter cada vez menos informação, convergindo para zero.

Definição Qualquer conjunto de subespaços vetoriais $\{V_{2^j}\}$ que satisfaçam as propriedades 4.1-4.6 é chamado de *Aproximação por Multiresolução em $L^2(R)$*. O conjunto de operadores $\{A_{2^j}\}$associados a este conjunto de espaços vetoriais gera as aproximações de qualquer função $f(x) \in L^2(R)$ em todos os níveis de resolução 2^j, para $j \in Z$.

Teorema A Seja $\{V_{2^j}, j \in Z\}$ uma aproximação por multiresolução em $L^2(R)$. Dado *j*, há um único operador A_{2^j} que faz a projeção ortogonal de *f(x)* na base ortonormal de V_{2^j}. Existe uma única função $\phi(x) \in L^2(R)$ responsável pela formação de todas as bases ortonormais de todos os subespaços. Essa função $\phi(x)$ é denominada função-escala que define [10]:

$$\phi_{j,n}(x) = 2^{j/2} \phi(2^j x - n) \qquad , \quad n \in Z \tag{4.7}$$

e essa família, através da variação de *n,* é uma base ortonormal para cada V_{2^j}. Essa família será referenciada ao longo do texto simplesmente como $\{\phi_{j,n}\}$.

A relação entre $\{\phi_{j,n}\}$ e V_{2^j} é ilustrada na Fig. 4.1, que exemplifica o caso para $j \in \{-1,0,1\}$. Combinando cada um desses valores de *j* com alguns valores de *n*, obtem-se as três bases ortonormais B_1, B_2 e B_3 a seguir, como exemplo.

$$\bullet\ j=1,\begin{cases}\vdots\\ n_1=0\Rightarrow\phi_{1,0}=\sqrt{2}\cdot\phi(2x)\\ n_2=1\Rightarrow\phi_{1,1}=\sqrt{2}\cdot\phi(2x-1)\\ n_3=2\Rightarrow\phi_{1,2}=\sqrt{2}\cdot\phi(2x-2)\\ \vdots\\ B_1\end{cases}\qquad \bullet\ j=0,\begin{cases}\vdots\\ n_1=0\Rightarrow\phi_{0,0}=\phi(x)\\ n_2=1\Rightarrow\phi_{0,1}=\phi(x-1)\\ n_3=2\Rightarrow\phi_{0,2}=\phi(x-2)\\ \vdots\\ B_2\end{cases}$$

$$\bullet\ j=-1,\begin{cases}\vdots\\ n_1=0\Rightarrow\phi_{-1,0}=1/\sqrt{2}\cdot\phi(2^{-1}x)\\ n_2=1\Rightarrow\phi_{-1,1}=1/\sqrt{2}\cdot\phi(2^{-1}x-1)\\ n_3=2\Rightarrow\phi_{-1,2}=1/\sqrt{2}\cdot\phi(2^{-1}x-2)\\ \vdots\\ B_3\end{cases} \tag{4.8}$$

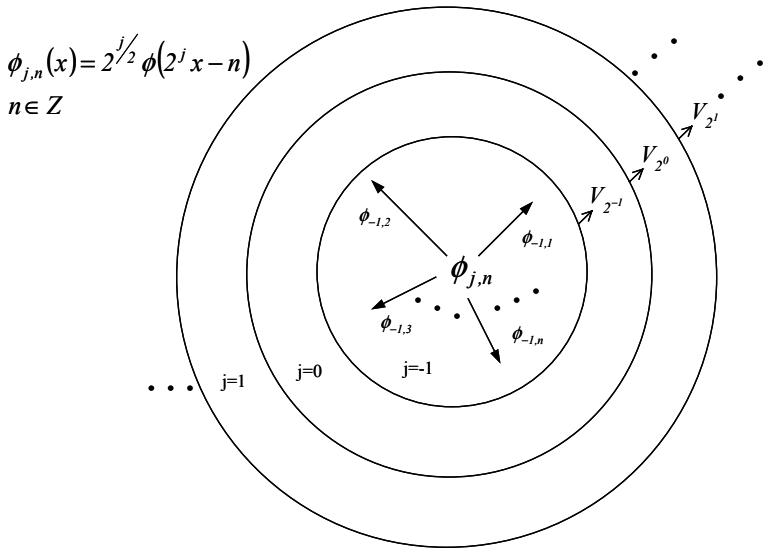


Fig. 4.1. Esquema ilustrativo da relação entre $\phi_{j,n}(x)$ e V_{2^j} para $j\in$ [-1,1]

As três bases respectivamente dos três subespaços vetoriais do exemplo, embora diferentes, são geradas pela mesma função-escala $\phi(x)$. Na Fig. 4.1 somente está representada a base ortonormal de $V_{2^{-1}}$.

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

$$\phi(x)=O(x^{-2})\qquad e\qquad |\phi'(x)|=O(x^{-2}) \tag{4.9}$$

A aproximação (projeção ortogonal de *f(x)*) no subespaço vetorial V_{2^j} pode ser obtida através da decomposição desta função na base ortonormal definida no Teorema A. Assim:

$$\forall f(x) \in L^2(R)\ ,\quad A_{2^j} \cdot f(x) = \sum_{n=-\infty}^{\infty} \langle f\ , \phi_{j,n} \rangle \cdot \phi_{j,n}(x) \tag{4.10}$$

e coeficientes dessa aproximação podem ser obtidos em função dos seguintes produtos internos:

$$A^d_{2^j} \cdot f = \{ \langle f\ , \phi_{j,n} \rangle\ , n \in Z \} \tag{4.11}$$

onde $A^d_{2^j} \cdot f$ é a aproximação discreta de *f(x)* na resolução 2^j [10].

4.2 - Implementação da Transformada de Multiresolução

Os sinais utilizados na prática são discretos, sendo simbolizados por $A^d_{2^j} \cdot f$ e possuindo resolução máxima finita. Por uma questão de normalização, será utilizada uma resolução máxima igual a 1 (i.e., um ponto por unidade de comprimento em *x*). Assim, $A^d_1 \cdot f$ é a aproximação discreta de *f(x)* nessa resolução.

A Propriedade 3 diz que a partir de $A^d_1 \cdot f$ pode-se calcular todas as aproximações discretas $A^d_{2^j} \cdot f$ para $j<0$ (em termos de teoria de filtros entende-se: a partir do sinal discreto original, podem ser obtidas todas as aproximações derivadas de sua subamostragem). Esta seção destina-se a descrever algoritmos simples e iterativos para o cálculo destas aproximações discretas [6][10].

Seja $\{ V_{2^j}, j \in Z \}$ uma aproximação por multiresolução e sua função-escala correspondente $\phi(x)$. De acordo com o Teorema A, a família $\{ \phi_{j,n} \}$ é uma base ortonormal para cada V_{2^j} [10][12]. Sabe-se também que $V_{2^j} \subset V_{2^{j+1}}$. Pode-se então expandir $\{ \phi_{j,n} \}$ na base ortonormal de $V_{2^{j+1}}$:

$$\phi_{j,n}(x) = \sum_{k=-\infty}^{\infty} \langle \phi_{j,n}\ , \phi_{j+1,k} \rangle\, \phi_{j+1,k}(x) \tag{4.12}$$

Fazendo a mudança de variáveis $u=2^{-j-1}y+n/2^j$ no produto interno de (4.12), tem-se:

$$\langle \phi_{j,n}\ , \phi_{j+1,k} \rangle = \langle 2^{j/2}\, \phi(2^j u - n)\ ,\ 2^{(j+1)/2}\, \phi(2^{j+1} u - k) \rangle =$$
$$2^{j/2+(j+1)/2} \int_{-\infty}^{\infty} \phi(2^{-1} y) \cdot \phi(y - (k-2n)) 2^{-j-1}\, dy = 2^{-1/2} \langle \phi(2^{-1} u), \phi(u - (k-2n)) \rangle \tag{4.13}$$

Assim, retornando à (4.12) tem-se:

$$\phi_{j,n}(x) = \sum_{k=-\infty}^{\infty} \langle \phi_{-1,0}(u)\ ,\ \phi_{0,k-2n}(u) \rangle \cdot \phi_{j+1,k}(x) \tag{4.14}$$

Sabe-se que todas as aproximações do sinal discreto original podem ser obtidas através de diferentes taxas de subamostragem do mesmo. O objetivo desse estudo é, contudo, obter aproximações do sinal discreto original a partir de outras aproximações, que será a base da análise multiresolução. No caso, o objetivo é obter a projeção em 2^j a partir da projeção em 2^{j+1}.

Assim, calculando cada coeficiente resultante da projeção de *f(x)* através da aplicação do produto interno de *f(x)* com ambos os lados de (4.14), tem-se:

$$\left\langle f, \phi_{j,n} \right\rangle = \sum_{k=-\infty}^{\infty} \left\langle \phi_{-1,0}(u),\ \phi_{0,k-2n}(u) \right\rangle \left\langle f(u), \phi_{j+1,k}(u) \right\rangle \tag{4.15}$$

Definindo o filtro discreto *h(m)* de resposta impulsiva em (4.16), pode-se encarar (4.15) como um processo de filtragem.

$$h(m) = \left\langle \phi_{-1,0}(u),\ \phi_{0,m}(u) \right\rangle \quad \forall n \in Z \tag{4.16}$$

Assim pode-se reescrever (4.14) como sendo:

$$\phi_{j,n}(x) = \sum_{k=-\infty}^{\infty} h(k-2n) \cdot \phi_{j+1,k}(x) \tag{4.17}$$

Analogamente, (4.15) pode ser reescrita como sendo:

$$\left\langle f, \phi_{j,n} \right\rangle = \sum_{k=-\infty}^{\infty} h(k-2n) \cdot \left\langle f, \phi_{j+1,k} \right\rangle \tag{4.18}$$

que é a expressão que para *n* fixo obtém 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, (4.18) mostra que $A_{2^j}^d \cdot f$ pode ser calculada convoluindo $A_{2^{j+1}}^d \cdot f$ com *h(-n)* e dizimando o resultado por um fator de 2, conforme esquematizado no ramo superior da Fig. 4.2.

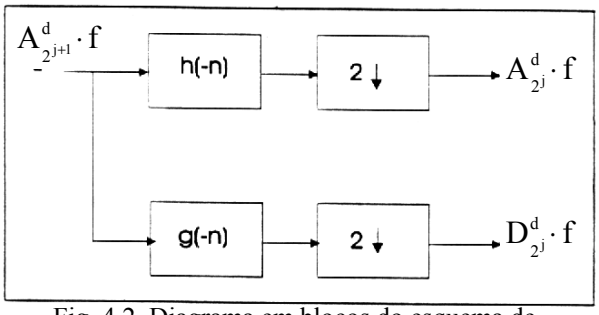


Fig. 4.2. Diagrama em blocos do esquema de decomposição.

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. Esse processamento é conhecido como

codificação piramidal [13]. Assim, supondo que a aproximação discreta de *f(x)* no maior nível de resolução seja completamente caracterizada por *N* coeficientes: $A_1^d \cdot f = \alpha_n$ para $1 \leq n \leq N$; cada aproximação discreta $A_{2^j}^d \cdot f$ para $j<0$ terá $2^j.N$ amostras (coeficientes).

Para obter a expressão que determina a função-escala $\phi(x)$ basta fazer $j= n= 0$ na equação (4.17), resultando em:

$$\phi(x) = \sum_{k=-\infty}^{\infty} h(k) \cdot 2^{1/2} \phi(2x-k) \tag{4.19}$$

A TF de (4.19) é obtida recursivamente da seguinte forma:

$$\boldsymbol{\phi}(\mathbf{w}) = \sum_{k=-\infty}^{\infty} h(k) \frac{\boldsymbol{\phi}(0{,}5w)}{2^{1/2}} e^{-i\,0{,}5wk} = \frac{\boldsymbol{\phi}(0{,}5w)}{2^{1/2}} H(0{,}5w) \tag{4.20}$$

$$\boldsymbol{\phi}(\mathbf{w}) = \boldsymbol{\phi}(0) \cdot \prod_{m=1}^{\infty} \frac{H\left(w \cdot 2^{-m}\right)}{2^{1/2}} \tag{4.21}$$

Porat em [14] coloca que para qualquer análise multiresolução, $\boldsymbol{\phi(0)=1}$. Isto implica que a família de funções $\{\phi_{j,n}\}$ não são *wavelets*, uma vez que não obedecem à condição de admissibilidade (equação (2.18)). Em [34] também mostra-se que se *h(k)* é não-nulo apenas para $0 \leq k \leq N-1$ (filtro FIR), então $\phi(x)$ é não-nula somente para $0 \leq x \leq N-1$. Deve-se ter em mente que o objetivo é determinar $\phi(x)$ tal que seja bem localizado em x. Assim, para obter essa $\phi(x)$ bem localizada deve-se impor um conjunto de restrições a *h(k)*. Sendo ***H(w)*** a TF de *h(n):*

$$\boldsymbol{H}(\boldsymbol{w}) = \sum_{n=-\infty}^{\infty} h(n) e^{-inw} \tag{4.22}$$

H(w) deve satisfazer as propriedades (4.23 - 4.26) [12]:

$$H(0) = \sum_{k=-\infty}^{\infty} h(k) = \sqrt{2} \text{ , obtida fazendo } w=0 \text{ em (4.21).} \tag{4.23}$$

$$h(n) = O\left(x^{-2}\right) \text{ quando } x \rightarrow \infty \tag{4.24}$$

$$|H(w)|^2 + |H(w+\pi)|^2 = 2 \text{ , } \forall w \text{ [14].} \tag{4.25}$$

$$|H(w)| \neq 0 \qquad para \;\; w \in \left[0, \pi/2\right] \tag{4.26}$$

Note-se que estas propriedades indicam que *H(w)* é um filtro passa-baixas.

4.3 - O Sinal de Detalhamento

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

$$W_{2^j} \text{ é ortogonal a } V_{2^j}$$
$$V_{2^j} \oplus W_{2^j} = V_{2^{j+1}} \tag{4.27}$$

conforme esquematizado na Fig. 4.3.

O sinal formado a partir da diferença de informação entre os níveis de resolução 2^j e 2^{j+1} é chamado de sinal de detalhamento no nível de resolução 2^j. Na subseção 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}.

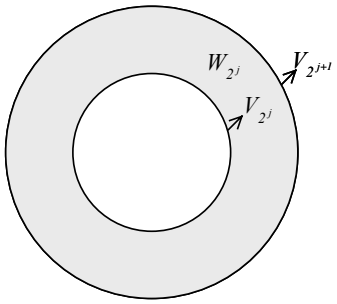


Fig. 4.3. O espaço vetorial W_{2^j}

Para calcular a projeção de *f(x)* em W_{2^j}, precisa-se encontrar, analogamente ao caso da seção anterior, uma base ortonormal para gerar o subespaço W_{2^j}. Dessa forma, precisa-se encontrar uma função $\psi(x)$ capaz de gerar a base ortogonal em questão. O objetivo desta seção é então assumir a existência de uma família $\left\{\psi_{j,n}(x) = 2^{j/2} \cdot \psi\left(2^j x - n\right), n \in Z\right\}$ e verificar se ela é uma base ortonormal para W_{2^j} [6][10]. Essa família será referenciada ao longo do texto simplesmente como $\left\{\psi_{j,n}\right\}$.

Assumindo a existência de $\psi(x)$, para qualquer $n \in Z$, a função $\psi_{j,n}(x) = 2^{j/2} \cdot \psi\left(2^j x - n\right)$ pertence a $W_{2^j} \subset V_{2^{j+1}}$. Pode-se então expandir $\psi_{j,n}$ na base ortonormal de $V_{2^{j+1}}$:

$$\psi_{j,n}(x) = \sum_{k=-\infty}^{\infty} \left\langle \psi_{j,n} , \phi_{j+1,k} \right\rangle \cdot \phi_{j+1,k}(x) \tag{4.28}$$

Fazendo a mudança de variáveis no produto interno de (4.28), tem-se analogamente à seção anterior:

$$\left\langle \psi_{\mathrm{j,n}} , \phi_{\mathrm{j+1,k}} \right\rangle = 2^{j/2+(j+1)/2} \int_{-\infty}^{\infty} \psi\left(2^{-1} y\right) \cdot \phi\left(y - (k-2n)\right) 2^{-j-1} dy = 2^{-1/2} \left\langle \psi\left(2^{-1} u\right), \phi\left(u - (k-2n)\right) \right\rangle \tag{4.29}$$

Assim, retornando à (4.28) tem-se:

$$\psi_{j,n}(x) = \sum_{k=-\infty}^{\infty} \left\langle \psi_{-1,0}(u) \ , \ \phi_{0,k-2n}(u) \right\rangle \cdot \phi_{j+1,k}(x) \tag{4.30}$$

Calculando-se cada coeficiente resultante da projeção de *f(x)* através da aplicação do produto interno de *f(x)* com ambos os lados de (4.28), tem-se:

$$\left\langle f , \psi_{j,n} \right\rangle = \sum_{k=-\infty}^{\infty} \left\langle \psi_{-1,0}(u), \ \phi_{0,k-2n}(u) \right\rangle \cdot \left\langle f(u), \phi_{j+1,k}(u) \right\rangle \tag{4.31}$$

Novamente aplicando a teoria de filtros, definindo o filtro discreto *g(n)* de resposta impulsiva em (4.32), pode-se encarar (4.31) como um processo de filtragem.

$$g(m) = \left\langle \psi_{-1,0}(u), \ \phi_{0,m}(u) \right\rangle \quad \forall n \in Z \tag{4.32}$$

Assim pode-se reescrever (4.30) como sendo:

$$\psi_{j,n}(x) = \sum_{k=-\infty}^{\infty} g(k-2n) \cdot \phi_{j+1,k}(x) \tag{4.33}$$

Analogamente, (4.31) pode ser reescrita como sendo:

$$\left\langle f , \psi_{j,n} \right\rangle = \sum_{k=-\infty}^{\infty} g(k-2n) \cdot \left\langle f , \phi_{j+1,k} \right\rangle \tag{4.34}$$

Para se obter a expressão que determina a função-escala $\psi(x)$ basta fazer $j = n = 0$ na equação (4.33), resultando em:

$$\psi(x) = \sum_{k=-\infty}^{\infty} g(k) \cdot 2^{1/2} \phi(2x-k) \tag{4.35}$$

Tomando-se a TF de ambos os lados de (4.33), tem-se:

$$\psi(w) = \sum_{k=-\infty}^{\infty} g(k) \frac{\phi(0{,}5w)}{2^{1/2}} e^{-i\,0{,}5w\,n} = \frac{\phi(0{,}5w)}{2^{1/2}} G(0{,}5w) \tag{4.36}$$

Deve-se notar que, diferentemente de $\phi(w)$, não há recursão para $\psi(w)$. $G(w)$ deve satisfazer as propriedades (4.37), (4.38) e (4.40):

$$|G(w)|^2 + |G(w+\pi)|^2 = 2 \ , \forall w \tag{4.37}$$

[14].

Porat [14] também coloca que *G(w)* e *H(w)* satisfazem à seguinte relação:

$$H(w)G^*(w)+H(w+\pi)G^*(w+\pi)=0 \tag{4.38}$$

Testando por fim, a condição de admissibilidade para $\psi(w)$, tem-se primeiramente substituindo (4.23) em (4.25):

$$|H(0)|^2+|H(\pi)|^2=2+|H(\pi)|^2=2\Rightarrow|H(\pi)|^2=0 \tag{4.39}$$

e em seguida substituindo (4.23) e (4.39) em (4.38):

$$G(0)=0\Rightarrow\psi(0)=\frac{\phi(0)}{2^{1/2}}\cdot G(0)\ \Rightarrow\psi(0)=0 \tag{4.40}$$

Logo, $\psi(w)$ satisfaz a condição de admissibilidade. Assim, a família $\{\psi_{j,n}\}$ é uma base *wavelet*.

Resta agora concatenar algumas equações anteriores para se obter a relação entre os filtros *g(n)* e *h(n)* que será a base da análise multiresolução. Para isso, primeiro definindo:

$$H_a=H(w)\ \ ,\ \ G_a=G(w)\ \ ,\ \ H_b=H(w+\pi)\ \ ,\ \ G_b=H(w+\pi) \tag{4.41}$$

e em seguida substituindo (4.41) em (4.38), tem-se:

$$G_a/G_b=-\left(H_b/H_a\right)^* \tag{4.42}$$

Dividindo-se o primeiro lado de (4.42) 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:

$$\frac{G_a/G_b}{\left[1+\left|G_a/G_b\right|^2\right]^{1/2}}=-\frac{\left(H_b/H_a\right)^*}{\left[1+\left|H_b/H_a\right|^2\right]^{1/2}} \tag{4.43}$$

Multiplicando matematicamente e substituindo (4.25) e (4.37) em (4.43), obtém-se:

$$G_a=-\frac{|H_a|G_b}{|G_b|H_a^*}\cdot H_b^*=-A(w)\cdot H_b^* \tag{4.44}$$

onde *A(w)* é uma função passa-tudo, uma vez que $|A(w)|=1,\ \forall w$. Conseqüentemente $|G_a|=|H_b|$, que é a primeira relação entre os filtros e nos diz que a magnitude do espectro de *g(n)* é igual ao espectro de *h(n)* deslocado de π. Logo, *G(w)* é um filtro passa-alta.

Para analisar agora a questão da fase dos filtros, basta substituir (4.44) em (4.38):

$$\begin{aligned} H(w)\left[-A(w)H^*(w+\pi)\right]^* \; + \; H(w+\pi)\left[-A(w+\pi)H^*(w+2\pi)\right]^* = 0 \\ A(w)+A(w+\pi)=0 \end{aligned} \tag{4.45}$$

Deve-se notar que a condição de admissibilidade está embutida em (4.44) (4.45). Freqüentemente se escolhe $A(w)=e^{-iw}$, escolha que obedece à $|A(w)|=1$ e à (4.45). Substituindo essa escolha em (4.44) tem-se:

$$\begin{aligned} G(w) &= e^{-iw}\cdot H^*(w+\pi) \\ g(n) &= (-1)^{n-1}\cdot h^*(1-n) \end{aligned} \tag{4.46}$$

A equação (4.46) dá a relação entre os filtros *g(n)* e *h(n)* utilizados no processo de decomposição e síntese da TW [6]. Qualquer par de filtros que obedeça às condições apresentadas nesta seção pode ser utilizado, mas é usual escolher *h(n)* como sendo um filtro passa-baixas. Sendo *h(n)* um filtro passa-baixas, *g(n)* é automaticamente um filtro passa-altas para esse procedimento. Note-se que determinando *g(n)* e *h(n)*, são determinadas $\psi(x)$ e $\phi(x)$.

Com base na análise feita, pode-se enunciar o seguinte teorema:

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

$$\boldsymbol{\psi}(\boldsymbol{w}) = G(0{,}5w)\,\phi(0{,}5w)\,2^{-1/2} \tag{4.47}$$

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

$$\psi_{j,n}(x)=2^{j/2}\cdot\psi(2^j x-n)\;,\; n\in Z \tag{4.48}$$

chamada no texto de $\{\psi_{j,n}\}$ é uma base ortonormal em W_{2^j} [6].

Conforme colocado anteriormente, o objetivo desta seção era assumir a existência de uma família $\{\psi_{j,n}\}$ e verificar se ela era uma base ortonormal para W_{2^j}, o que foi confirmado com a análise feita e formalizado no Teorema B.

O Teorema B 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 (4.23)-(4.26), pode-se encontrar a função-escala e a *wavelet*-mãe a partir de (4.21) e (4.47), 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 [5].

Assim, similarmente à seção anterior, seja D_{2^j} o operador de projeção ortogonal no subespaço vetorial W_{2^j}. O sinal de detalhamento de *f(x)*, $D_{2^j} \cdot f(x)$, com resolução 2^j é dado por:

$$D_{2^j} \cdot f(x) = \sum_{n=-\infty}^{\infty} \langle f(u), \psi_{j,n}(u) \rangle \cdot \psi_{j,n}(x) \tag{4.49}$$

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

$$D_{2^j}^d \cdot f = \{ \langle f, \psi_{j,n} \rangle , n \in Z \} \tag{4.50}$$

onde $D_{2^j}^d \cdot f$ contém a diferença de informação entre $A_{2^{j+1}}^d \cdot f$ e $A_{2^j}^d \cdot f$

De (4.34) nota-se que é possível calcular o sinal de detalhamento $D_{2^j}^d \cdot f$ através da convolução de $A_{2^{j+1}}^d \cdot f$ com o filtro discreto *g(-n)*, e aplicando dizimação por um fator de dois no resultado, conforme esquematizado no ramo inferior da Fig. 4.2. O processo completo é definido por (4.11) e (4.49).

4.4 - Implementação da Representação *Wavelet* Ortogonal

A representação *wavelet* ortogonal do sinal discreto $A_1^d \cdot f$ pode ser calculada através de sucessivas decomposições de $A_{2^{j+1}}^d \cdot f$ em $A_{2^j}^d \cdot f$ e $D_{2^j}^d \cdot f$, sendo $-J \le j \le -1$:

$$\left(A_{2^{-J}}^d \cdot f , \{ D_{2^j}^d \cdot f \}_{-J \le j \le -1} \right) \tag{4.51}$$

ou seja, composta pela aproximação $A_{2^{-J}}^d \cdot f$ com resolução mais "grosseira" ("DC") e pelos detalhes $D_{2^j}^d \cdot f$ com resoluções 2^j ("ACs"), sendo $-J \le j \le -1$. 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* (4.51) apresenta o mesmo número de amostras que $A_1^d \cdot f$, graças à ortogonalidade da representação.

Toda essa análise, feita nas Seções 4.1- 4.4, detalha o algoritmo de Mallat cujo princípio de funcionamento havia sido brevemente colocado na Seção 3.2.

4.5 – Reconstrução a partir da Representação *Wavelet* Ortogonal

Foi visto anteriormente que o espaço vetorial definido para a representação *wavelet* ortogonal é completo. Como W_{2^j} é o complemento ortogonal de V_{2^j} em $V_{2^{j+1}}$, $\left(\{\phi_{j,n}(x)\},\{\psi_{j,n}(x)\}\right)_{n\in Z}$ é uma base ortonormal em $V_{2^{j+1}}$. Pode-se escrever $\phi_{j+1,n}(x)$ como:

$$\phi_{j+1,n}(x)=\sum_{k=-\infty}^{\infty}\langle\phi_{j+1,n}\,,\phi_{j,k}\rangle\cdot\phi_{j,k}(x)+\sum_{k=-\infty}^{\infty}\langle\phi_{j+1,n}\,,\psi_{j,k}\rangle\cdot\psi_{j,k}(x) \tag{4.52}$$

Calculando-se o produto interno de ambos os lados de (4.52) com *f(x)*, tem-se:

$$\langle f(x),\phi_{j+1,n}(x)\rangle=\sum_{k=-\infty}^{\infty}\langle\phi_{j+1,n}\,,\,\phi_{j,k}\rangle\cdot\langle f(x),\phi_{j,k}(x)\rangle+\sum_{k=-\infty}^{\infty}\langle\phi_{j+1,n}\,,\,\psi_{j,k}\rangle\cdot\langle f(x),\psi_{j,k}(x)\rangle \tag{4.53}$$

Em seguida, fazendo-se a mudança de variáveis em (4.53), obtém-se:

$$\langle\phi_{j+1,n}(u),\phi_{j,k}(u)\rangle=\langle\phi_{-1,0}(u),\phi_{0,n-2k}(u)\rangle=h(n-2k) \tag{4.54}$$

$$\langle\phi_{j+1,n}(u),\psi_{j,k}(u)\rangle=\langle\psi_{-1,0}(u),\phi_{0,n-2k}(u)\rangle=g(n-2k) \tag{4.55}$$

que, se substituídas em (4.53), permitem que ela seja reescrita em função dos filtros *g(n)* e *h(n)*:

$$\underbrace{\langle f(x),\phi_{j+1,n}(x)\rangle}_{\text{componente de } A^d_{2^{j+1}}\cdot f}=\sum_{k=-\infty}^{\infty}h(n-2k)\cdot\underbrace{\langle f(x),\phi_{j,k}(x)\rangle}_{\text{componente de } A^d_{2^j}\cdot f}+\sum_{k=-\infty}^{\infty}g(n-2k)\cdot\underbrace{\langle f(x),\psi_{j,k}(x)\rangle}_{\text{componente de } D^d_{2^j}\cdot f} \tag{4.56}$$

Em (4.56), 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. 4.4.

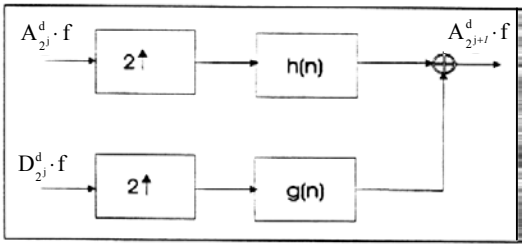


Fig. 4.4. Esquema de reconstrução do sinal a partir da representação wavelet.

Comparando as Figs. 4.2 e 4.4, pode-se observar que no algoritmo de decomposição utiliza-se *h-(n)* e *g(-n)*, enquanto na síntese utiliza-se filtros de resposta impulsiva *h(n)* e *g(n)*.

4.6 – Transformada *Wavelet* Discreta (TWD): Implementações

O objetivo desta subseção é conectar alguns conceitos colocados nas seções anteriores, de forma a permitir uma visualização global da aplicação da transformada *wavelet* discreta (TWD). A TWD pode ser implementada diretamente [5]:

$$W_{j,n} = \sum_i f(i\Delta t) \cdot \psi_{j,n}(i\Delta t) \tag{4.57}$$

$$f(i\Delta t) = \sum_{j,n} W_{j,n} \cdot \psi_{j,n}(i\Delta t) \tag{4.58}$$

Nessa implementação, basta colocar a *wavelet*-básica (que obedeça a condição de admissibilidade) e variar *j* e *n*, não havendo necessidade de se fazer o cálculo de *h(n), g(n)* ou da função-escala.

Contudo, a escolha arbitrária, mesmo seguindo o procedimento descrito, pode levar a *wavelets* sem o comportamento desejado. Nesse contexto, diversos pesquisadores se aprofundaram na construção de *wavelets* que obedecessem à determinadas características importantes para as mais diversas aplicações, como será abordado no Capítulo 7. De forma geral, esses pesquisadores elaboraram algoritmos partindo de *h(n)* e achando $\phi(x)$, *g(n)* e $\psi(x)$ direcionando as características.

Uma vez determinada a *wavelet*-mãe do sistema, pode-se alternativamente implementar a TWD através do algoritmo rápido de *Mallat* (FWT) dado na Seção 3.2, que é computacionalmente mais eficiente [5]. Deve ser ressaltado que as condições a respeito de *h(n)* e *g(n)* colocadas anteriormente devem ser satisfeitas. O interessante acerca do algoritmo de *Mallat* é que ele não manipula as *wavelets* e funções-escala explicitamente, mas sim, as projeções de *f(x)* na hierarquia de subespaços. Veja, o filtro *h(-n)* via equação (4.17) mapeia a projeção de *f(x)* no subespaço $V_{2^{j+1}}$ para a projeção no subespaço V_{2^j}. Similarmente, o filtro *g(-n)* via (4.34) mapeia a projeção de *f(x)* no subespaço $V_{2^{j+1}}$ para a projeção no subespaço W_{2^j}. Assim, as saídas do esquema da Fig. 3.5 podem ser vistas em (4.51).

4.7 – Comentários

Este capítulo apresentou a implementação da TWD (decomposição e síntese) para o caso ortogonal. A importância fundamental desse capítulo reside na geração dos sinais PB 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 5

Biortogonalidade

Muitas propriedades interessantes podem ser obtidas ao se utilizar uma base ortogonal para a expansão de sinais. Através dela o sinal pode ser decomposto em um conjunto de coeficientes independentes via projeção, onde cada coeficiente corresponde a um elemento dessa base. Nesse tipo de representação, não há redundância.

No processamento de imagens, contudo, é importante que a transformada utilizada apresente outras vantagens tais como suporte compacto, simetria e regularidade. Wavelets de suporte compacto implicam no uso de filtros FIR e, conseqüentemente, algoritmos de rápida implementação. Filtros simétricos (também chamados de filtros fase linear) tornam possível o cascateamento de bancos de filtros sem a necessidade de compensação de fase. A regularidade está diretamente relacionada com o número de funções possíveis de serem representadas pela TW com maior fidelidade. Contudo, suporte compacto é conflitante com a regularidade, conforme será abordado na Seção 7.2 [9][15][16].

Além disso, teoricamente, os filtros adotados devem garantir a reconstrução do sinal sem erros. Infelizmente, Daubechies *et.al* [15] afirma que não há filtro FIR ortonormal e simétrico trivial com propriedade de reconstrução exata. Os únicos filtros FIR simétricos com propriedade de reconstrução exata são os filtros correspondentes à *wavelet* de Haar [9], que por ser descontínua, possui má localização em freqüência.

Para contornar essas questões conflitantes, a condição de ortogonalidade é relaxada, gerando a biortogonalidade. Antes de expô-la porém, se faz necessário abordar a teoria de *frames*.

5.1 – Teoria de *Frames*

As *wavelets* biortogonais são uma generalização das *wavelets* ortogonais [17]-[19]. Foi colocado em (4.7) que a família $\{\phi_{j,n}\}$é uma base ortonormal para V_{2^j}. Essa família também constitui um *frame*, sendo a definição de *frame* dada por:

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

$$A\|f\|^2 \leq \sum_{k\in K} |\langle g_k, f\rangle|^2 \leq B\|f\|^2 \quad , \forall f \in L^2(R) \tag{5.1}$$

onde *A* e *B* são chamados de limitantes do *frame* [5]. O *frame* é um conjunto de funções, tais que qualquer função *f(x)* não-nula deve ter uma projeção não-nula em pelo menos uma delas.

Um *frame* é necessariamente um conjunto completo em $L^2(R)$, mas as funções de um *frame* podem ser linearmente dependentes (LD, implicando em redundância) ou linearmente independentes (LI, sem redundância). Nem todo *frame* é uma base em $L^2(R)$. Caso as funções sejam LD, o *frame* não constitui uma *base* para $L^2(R)$. Caso sejam LI, o *frame* constitui uma base e é chamado de *frame* exato. Nesse caso, 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:

$$\vec{\alpha_1} = \vec{e_1} \qquad \vec{\alpha_2} = -\frac{1}{2}\vec{e_1} + \frac{\sqrt{3}}{2}\vec{e_2} \qquad \vec{\alpha_3} = -\frac{1}{2}\vec{e_1} - \frac{\sqrt{3}}{2}\vec{e_2} \tag{5.2}$$

tem-se:

$$\sum_{j=1}^{3}\left|\left\langle f, \vec{\alpha_j}\right\rangle\right|^2 = \left|\left\langle f, \vec{e_1}\right\rangle\right|^2 + \left|\left(-\frac{1}{2}\right)\left\langle f, \vec{e_1}\right\rangle + \left(\frac{\sqrt{3}}{2}\right)\left\langle f, \vec{e_2}\right\rangle\right|^2 + \left|\left(-\frac{1}{2}\right)\left\langle f, \vec{e_1}\right\rangle + \left(-\frac{\sqrt{3}}{2}\right)\left\langle f, \vec{e_2}\right\rangle\right|^2$$
$$= \frac{3}{2}\left|\left\langle f, \vec{e_1}\right\rangle\right|^2 + \frac{3}{2}\left|\left\langle f, \vec{e_2}\right\rangle\right|^2 = \frac{3}{2}\left|\left\langle f, \vec{e}\right\rangle\right|^2 \tag{5.3}$$

O conjunto de vetores $\left\{\vec{\alpha_1}, \vec{\alpha_2}, \vec{\alpha_3}\right\}$ obedece (5.1), logo compõem um *frame*. São também LD, logo, não constituem uma base. Como *A=B=3/2*, o *frame* é estreito, onde 3/2 é a razão de redundância [6].

Lema A 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 [9].

A qualquer *frame* $\{g_k, k \in K\}$ pode-se associar um "operador *frame*" T: $L^2(R) \rightarrow L^2(R)$ que realize a transformação expansão em série de *f(x)*:

$$T \cdot f(x) = \sum_{k \in K} \langle f, g_k \rangle \cdot g_k(x) \tag{5.4}$$

Porat [14] demonstrou que o operador *T* é inversível. Assim, preparando para definir a transformada inversa:

Lema B Define-se o conjunto de funções $\{\gamma_k, k \in K\}$ como sendo o *frame* dual de $\{g_k, k \in K\}$ obtido da seguinte forma [14]:

$$\gamma_k = T^{-1} g_k \tag{5.5}$$

Quando o *frame* é exato, os *frames* $\{\gamma_k\}$ e $\{g_k\}$ são biortogonais, logo:

$$\langle g_k, \gamma_l \rangle = \delta(k-l) \tag{5.6}$$

De (5.4) e (5.5) segue o Teorema C.

Teorema C 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:

$$f(x) = \sum_{k \in K} \langle f, \gamma_k \rangle \cdot g_k(x) \tag{5.7}$$

$$f(x) = \sum_{k \in K} \langle f, g_k \rangle \cdot \gamma_k(x) \tag{5.8}$$

As equações (5.7) e (5.8) mostram que pode-se expressar qualquer função de $L^2(R)$ 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 [8]. 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.

5.2 - *Wavelets* Ortogonais X *Wavelets* Biortogonais

Assim, estendendo para o contexto da TW, se a *wavelet* $\psi(x)$ gera um *frame* $\{\psi_{j,n}\}$, isto é:

$$A\|f\|^2 \le \sum_{j,k \in Z} \left|\langle f, \psi_{j,n} \rangle\right|^2 \le B\|f\|^2 \quad , \forall f \in L^2(R) \tag{5.9}$$

existe a inversa [9]. Assim, nesse contexto, definindo-se $\{\psi_{j,n}\}$ como sendo o *frame* dual de $\{\psi_{j,n}\}$; as equações (5.4), (5.8) e (5.7) ficam, respectivamente:

$$T \cdot f(x) = \sum_{j,n \in Z} \langle f, \psi_{j,n} \rangle \cdot \psi_{j,n}(x) \tag{5.10}$$

$$f(x) = T^{-1} \cdot T \cdot f(x) = \sum_{j,n \in Z} \langle f, \psi_{j,n} \rangle \cdot T^{-1} \cdot \psi_{j,n}(x) = \sum_{j,n \in Z} \langle f, \psi_{j,n} \rangle \cdot \tilde{\psi}_{j,n}(x) \tag{5.11}$$

$$f(x) = \sum_{j,n \in Z} \langle f, \tilde{\psi}_{j,n} \rangle \cdot \psi_{j,n}(x) \tag{5.12}$$

Quando $\{\psi_{j,n}\}$ é *frame* exato, os conjuntos $\{\psi_{j,n}\}$e$\{\tilde{\psi}_{j,n}\}$ são biortogonais, ou seja [17]:

$$\langle \psi_{j,k}, \tilde{\psi}_{l,m} \rangle = \int_{-\infty}^{\infty} \psi_{j,k}(x) \cdot \tilde{\psi}_{l,m}(x)\, dx = \delta(j-l, k-m) \qquad j,k,l,m \in Z \tag{5.13}$$

Caso os *frames* em questão sejam também estreitos, são ortogonais e (5.11) fica:

$$f(x)=\sum_{j,n\in Z}\left\langle f,\psi_{j,n}\right\rangle\cdot A^{-1}\psi_{j,n}(x)=A^{-1}\sum_{j,n\in Z}\left\langle f,\psi_{j,n}\right\rangle\cdot\psi_{j,n}(x) \tag{5.14}$$

Caso os *frames* obedeçam ao Lema A formando bases ortonormais, tem-se $\psi_{j,n}(x)=\tilde{\psi}_{j,n}(x)$ e:

$$f(x)=\sum_{j,n\in Z}\left\langle f,\psi_{j,n}\right\rangle\cdot\psi_{j,n}(x) \tag{5.15}$$

que foi a equação colocada em (2.29). Nota-se que fazendo $\tilde{\psi}_{j,n}(x)=\psi_{j,n}(x)$ em (5.12) obtém-se (5.15) evidenciando que, de fato, as *wavelets* ortogonais são um caso especial das *wavelets* biortogonais. As principais diferenças entre ambas são [6]:

- Os filtros ortogonais possuem necessariamente o mesmo comprimento, que deve ser par. Essa restrição não é imposta aos filtros biortogonais.
- A biortogonalidade permite que *wavelets* e funções-escala simétricas e de reconstrução exata sejam construídas, o que se constitui numa de suas principais vantagens, conforme será abordado no Capítulo 7.
- A identidade de Parseval não é válida para os sistemas biortogonais, ou seja, a norma dos coeficientes não é idêntica à norma da função, o que se constitui numa das principais desvantagens das *wavelets* biortogonais. Isso significa que dada a energia do sinal original, não é possível prever a energia da saída.

5.3 - Análise de Multiresolução para *Wavelets* Biortogonais

A análise de multiresolução para *wavelets* biortogonais é similar à apresentada para o caso ortogonal, porém com uma implementação ligeiramente mais complicada. No caso biortogonal há dois espaços vetoriais $\{V_{2^j}\}$ 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 $\{\tilde{W}_{2^j}\}$, bem como um par de *wavelets* duais $\psi(x)$ e $\tilde{\psi}(x)$ [5].

Os subespaços V_{2^j} e $\tilde{V}_{2^j}$ obedecem às propriedades da seção 5.1. Assim:

$$\begin{aligned}&\ldots\subset V_{2^{-1}}\subset V_{2^0}\subset V_{2^1}\subset V_{2^2}\subset\ldots\\&\ldots\subset \tilde{V}_{2^{-1}}\subset \tilde{V}_{2^0}\subset \tilde{V}_{2^1}\subset \tilde{V}_{2^2}\subset\ldots\end{aligned} \tag{5.16}$$

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 $\tilde{V}_{2^j}$. Essas bases são construídas a partir de deslocamentos, dilatações e

compressões das funções-escala $\phi(x)$ e $\widetilde{\phi}(x)$, respectivamente. As funções-escala obedecem a:

$$\left\langle \phi_{0,0} , \widetilde{\phi}_{0,m} \right\rangle = \delta(m) \tag{5.17}$$

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

$$\begin{aligned} V_{2^{j+1}} &= V_{2^j} \oplus W_{2^j} \qquad V_{2^j} \text{ não é ortogonal à } W_{2^j} \\ \widetilde{V}_{2^{j+1}} &= \widetilde{V}_{2^j} \oplus \widetilde{W}_{2^j} \qquad \widetilde{V}_{2^j} \text{ não é ortogonal à } \widetilde{W}_{2^j} \end{aligned} \tag{5.18}$$

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

$$\begin{aligned} &\ldots \subset W_{2^{-1}} \subset W_{2^0} \subset W_{2^1} \subset W_{2^2} \subset \ldots \\ &\ldots \subset \widetilde{W}_{2^{-1}} \subset \widetilde{W}_{2^0} \subset \widetilde{W}_{2^1} \subset \widetilde{W}_{2^2} \subset \ldots \end{aligned} \tag{5.19}$$

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

$$\left\langle \psi_{0,0} , \widetilde{\psi}_{0,m} \right\rangle = \delta(m) \tag{5.20}$$

Daubechies [9] também coloca a seguinte relação:

$$V_{2^j} \perp \widetilde{W}_{2^j} \quad e \quad \widetilde{V}_{2^j} \perp W_{2^j} \tag{5.21}$$

Para determinar qualquer um dos subespaços de (5.21), só é preciso que se conheça um deles. A equação (5.21) implica em:

$$\left\langle \phi_{0,0} , \widetilde{\psi}_{0,m} \right\rangle = \left\langle \widetilde{\phi}_{0,0} , \psi_{0,m} \right\rangle = \delta(m) \tag{5.22}$$

Sejam $\{\phi_{j,n}\}$ e $\{\widetilde{\phi}_{j,n}\}$ os dois *frames* duais geradores dos subespaços V_{2^j} e $\widetilde{V}_{2^j}$. De acordo com a teoria de *frames*, 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 é então dada por:

$$A_{2^j} \cdot f(x) = \sum_{n=-\infty}^{\infty} \left\langle f , \widetilde{\phi}_{j,n} \right\rangle \cdot \phi_{j,n}(x) \tag{5.23}$$

e sua aproximação discreta :

$$A_{2^j}^d \cdot f = \left\{ \left\langle f , \widetilde{\phi}_{j,n} \right\rangle , n \in Z \right\} \tag{5.24}$$

Analogamente, sendo $\{\psi_{j,n}\}$ e $\{\widetilde{\psi}_{j,n}\}$ os dois *frames* duais geradores dos subespaços W_{2^j}

e $\widetilde{W}_{2^j}$, 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:

$$D_{2^j} \cdot f(x) = \sum_{n=-\infty}^{\infty} \langle f, \widetilde{\psi}_{j,n} \rangle \cdot \psi_{j,n}(x) \tag{5.25}$$

e o sinal de detalhamento discreto é:

$$D_{2^j}^d \cdot f = \{ \langle f, \widetilde{\psi}_{j,n} \rangle \quad , n \in Z \} \tag{5.26}$$

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

$$\widetilde{\phi}_{j,n}(x) = \sum_{k=-\infty}^{\infty} \langle \widetilde{\phi}_{j,n}, \phi_{j+1,k} \rangle \cdot \widetilde{\phi}_{j+1,k}(x) = \sum_{k=-\infty}^{\infty} \langle \widetilde{\phi}_{-1,0}(u), \phi_{0,k-2n}(u) \rangle \cdot \widetilde{\phi}_{j+1,k}(x) = \sum_{k=-\infty}^{\infty} \widetilde{h}(k-2n) \cdot \widetilde{\phi}_{j+1,k}(x) \tag{5.27}$$

onde o filtro discreto de resposta impulsiva em questão é dado em (5.28):

$$\widetilde{h}(m) = \langle \widetilde{\phi}_{-1,0}(u), \phi_{0,m}(u) \rangle \tag{5.28}$$

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

$$\widetilde{\psi}_{j,n}(x) = \sum_{k=-\infty}^{\infty} \langle \widetilde{\psi}_{j,n}, \phi_{j+1,k} \rangle \cdot \widetilde{\phi}_{j+1,k}(x) = \sum_{k=-\infty}^{\infty} \langle \widetilde{\psi}_{-1,0}(u), \phi_{0,k-2n}(u) \rangle \cdot \widetilde{\phi}_{j+1,k}(x) = \sum_{k=-\infty}^{\infty} \widetilde{g}(k-2n) \cdot \widetilde{\phi}_{j+1,k}(x) \tag{5.29}$$

onde o filtro discreto de resposta impulsiva é dado em (5.30):

$$\widetilde{g}(m) = \langle \widetilde{\psi}_{-1,0}(u), \phi_{0,m}(u) \rangle \tag{5.30}$$

Convoluindo (5.27) e (5.29) com a função *f(x)*, tem-se respectivamente:

$$\langle f, \widetilde{\phi}_{j,n} \rangle = \sum_{k=-\infty}^{\infty} \widetilde{h}(k-2n) \cdot \langle f, \widetilde{\phi}_{j+1,k} \rangle \tag{5.31}$$

$$\langle f, \widetilde{\psi}_{j,n} \rangle = \sum_{k=-\infty}^{\infty} \widetilde{g}(k-2n) \cdot \langle f, \widetilde{\phi}_{j+1,k} \rangle \tag{5.32}$$

As equações (5.31) e (5.32) caracterizam o processo de decomposição da análise multiresolução biortogonal. Pode-se notar que as aproximações discretas $A_{2^j}^d \cdot f$ e $D_{2^j}^d \cdot f$ podem ser calculadas através da convolução de $A_{2^{j+1}}^d \cdot f$ com os filtros $\widetilde{h}(-n)$ e $\widetilde{g}(-n)$ respectivamente, seguidas pela dizimação por um fator de 2 (descarte de uma amostra sim e outra não do resultado).

5.4 - Reconstrução a partir da Representação *Wavelet* Biortogonal

O subespaço W_{2^j} é o complemento não-ortogonal de V_{2^j} em $V_{2^{j+1}}$. Portanto, o espaço $V_{2^{j+1}}$ pode ser gerado pela base $\left(\{\phi_{j,n}\},\{\psi_{j,n}\}\right)_{n\in Z}$. Como:

$$A^d_{2^{j+1}}\cdot f(x)=\sum_n \left\langle f(x),\tilde{\phi}_{j+1,n}\right\rangle \phi_{j+1,n} \tag{5.33}$$

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

$$\begin{aligned}\tilde{\phi}_{j+1,n}(x)&=\sum_{k=-\infty}^{\infty}\left\langle \tilde{\phi}_{j+1,n}(u),\phi_{j,k}(u)\right\rangle\cdot\tilde{\phi}_{j,k}(x)+\sum_{k=-\infty}^{\infty}\left\langle \tilde{\phi}_{j+1,n}(u),\psi_{j,k}(u)\right\rangle\cdot\tilde{\psi}_{j,k}(x)\\&=\sum_{k=-\infty}^{\infty}\left\langle \tilde{\phi}_{0,n-2k}(u),\phi_{-1,0}(u)\right\rangle\cdot\tilde{\phi}_{j,k}(x)+\sum_{k=-\infty}^{\infty}\left\langle \tilde{\phi}_{0,n-2k}(u),\psi_{-1,0}(u)\right\rangle\cdot\tilde{\psi}_{j,k}(x)\end{aligned} \tag{5.34}$$

Definindo os filtros discretos de resposta impulsiva em (5.35) e (5.36), pode-se encarar (5.34) como o processo de filtragem dado em (5.37).

$$h(m)=\left\langle \phi_{-1,0}(u),\tilde{\phi}_{0,m}(u)\right\rangle \tag{5.35}$$

$$g(m)=\left\langle \psi_{-1,0}(u),\tilde{\phi}_{0,m}(u)\right\rangle \tag{5.36}$$

$$\tilde{\phi}_{j+1,n}(x)=\sum_{k=-\infty}^{\infty}h(n-2k)\cdot\tilde{\phi}_{j,k}(x)+\sum_{k=-\infty}^{\infty}g(n-2k)\cdot\tilde{\psi}_{j,k}(x) \tag{5.37}$$

Convoluindo (5.37) com a função *f(x)*, tem-se:

$$\underbrace{\left\langle f(x),\tilde{\phi}_{j+1,n}(x)\right\rangle}_{\text{componente de } A^d_{2^{j+1}}\cdot f}=\sum_{k=-\infty}^{\infty}h(n-2k)\cdot\underbrace{\left\langle f(x),\tilde{\phi}_{j,k}(x)\right\rangle}_{\text{componente de } A^d_{2^j}\cdot f}+\sum_{k=-\infty}^{\infty}g(n-2k)\cdot\underbrace{\left\langle f(x),\tilde{\psi}_{j,k}(x)\right\rangle}_{\text{componente de } D^d_{2^j}\cdot f} \tag{5.38}$$

Em (5.38), nota-se que $A^d_{2^{j+1}}\cdot f$ pode ser reconstruída 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 $D^d_{2^j}\cdot f$. A reconstrução da análise de multiresolução biortogonal é caracterizada por:

$$A^d_{2^{j+1}}\cdot f(x)=A^d_{2^j}\cdot f(x)+D^d_{2^j}\cdot f(x) \tag{5.39}$$

O esquema da decomposição e síntese da TW biortogonal encontra-se na Fig. 5.1. Observa-se que na decomposição utiliza-se $\tilde{h}(-n)$ e $\tilde{g}(-n)$, enquanto na síntese utiliza-se filtros de resposta impulsiva *h(n)* e *g(n)*. Esta característica é diferente da TW ortogonal, na qual são utilizados os

mesmos filtros *h(n)* e *g(n)* na decomposição e na síntese. Além disso, os filtros correspondentes à *wavelet* biortogonal são mais flexíveis, fáceis de projetar e podem ser simétricos [15].

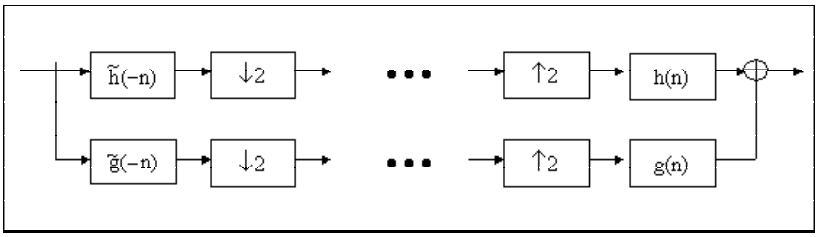


Fig. 5.1. Diagrama em blocos do algoritmo de decomposição e síntese da TWD biortogonal.

5.5 - Comentários

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 é apresentada a TWD bidimensional.

Capítulo 6

Análise Multiresolução Bidimensional

Já são conhecidos os excelentes resultados da TWD quando aplicada à compressão de imagens estáticas e vídeo. Bons exemplos podem ser encontrados em [20]-[31]. Este capítulo destina-se a estender a análise multiresolução para o caso 2D a partir de *wavelets* ortogonais [6][10]. O subespaço vetorial $V_{2^j}^2$ é definido como o produto tensorial dado por:

$$V_{2^j}^2 = V_{2^j}^1 \otimes V_{2^j}^1 \tag{6.1}$$

Sendo $\left\{V_{2^j}^1\right\}$ a aproximação por multiresolução em $L^2(R)$ dada na Seção 4.1, qualquer conjunto de subespaços vetoriais $\left\{V_{2^j}^2, j \in Z\right\}$ que satisfaçam as propriedades 1-6 da mesma subseção para o caso bidimensional é chamado de *Aproximação por Multiresolução em* $L^2(R^2)$. Ao longo do texto, esse conjunto será referenciado simplesmente por $\left\{V_{2^j}^2\right\}$, de forma a diferenciá-lo de cada $V_{2^j}^2$ em particular. A aproximação de uma imagem *f(x,y)* no nível de resolução 2^j é a sua projeção no subespaço vetorial $V_{2^j}^2$.

Estendendo o Teorema A 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 [10]. Assim, a família $\left\{\phi_{j;n_x,n_y}(x,y)\right\}$ dada por:

$$\phi_{j;n_x,n_y}(x,y) = 2^j \phi\left(2^j x - n_x\,,\, 2^j y - n_y\right) \qquad ,(n_x\,,n_y) \in Z^2 \tag{6.2}$$

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

Sendo $\phi(x)$ a função-escala unidimensional referente a $\left\{V_{2^j}^1\right\}$, Mallat [21] mostra o caso em que a função-escala bidimensional $\phi(x,y)$ ortogonal é separável, podendo ser expressa por:

$$\phi(x,y) = \phi(x) \cdot \phi(y) \tag{6.3}$$

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

$$\phi_{j;n_x,n_y}(x,y) = \phi_{j,n_x}(x) \cdot \phi_{j,n_y}(y) \qquad ,(n_x\,,n_y) \in Z^2 \tag{6.4}$$

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

$$A_{2^j} \cdot f(x,y) = \sum_{n_x=-\infty}^{\infty} \sum_{n_y=-\infty}^{\infty} \left\langle f(u,v), \phi_{j;n_x,n_y}(u,v) \right\rangle \cdot \phi_{j;n_x,n_y}(x,y) \tag{6.5}$$

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

$$A_{2^j}^d \cdot f = \left\{ \left\langle f(u,v), \phi_{j,n_x}(u) \cdot \phi_{j,n_y}(v) \right\rangle \qquad ,(n_x,n_y) \in Z^2 \right\} \tag{6.6}$$

6.1 – 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 $V_{2^{j+1}}^2$:

$$\phi_{j;n_x,n_y}(x,y) = \sum_{k_y=-\infty}^{\infty} \sum_{k_x=-\infty}^{\infty} \left\langle \phi_{j;n_x,n_y}(u,v), \phi_{j+1;k_x,k_y}(u,v) \right\rangle \cdot \phi_{j+1;k_x,k_y}(x,y) \tag{6.7}$$

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:

$$\left\langle f(x,y) \cdot \phi_{j;n_x,n_y}(x,y) \right\rangle = \sum_{k_y=-\infty}^{\infty} \sum_{k_x=-\infty}^{\infty} \left\langle \phi_{-1;0,0}(u,v), \phi_{0;(k_x-2n_x),(k_y-2n_y)}(u,v) \right\rangle \cdot \left\langle f(x,y) \cdot \phi_{j+1;k_x,k_y}(x,y) \right\rangle \tag{6.8}$$

Definindo o filtro discreto *h(l,m)* de resposta impulsiva em (6.9), pode-se encarar (6.8) como o processo de filtragem descrito em (6.10).

$$h(\ell,m) = \left\langle \phi_{-1;0,0}(u,v), \phi_{0;\ell,m}(u,v) \right\rangle = \left\langle \phi_{-1,0}(u), \phi_{0,\ell}(u) \right\rangle \cdot \left\langle \phi_{-1,0}(v), \phi_{0,m}(v) \right\rangle = h(\ell) \cdot h(m) \tag{6.9}$$

$$\left\langle f(x,y) \cdot \phi_{j;n_x,n_y}(x,y) \right\rangle = \sum_{k_y=-\infty}^{\infty} h(k_y - 2n_y) \cdot \sum_{k_x=-\infty}^{\infty} h(k_x - 2n_x) \cdot \left\langle f(x,y) \cdot \phi_{j+1;k_x,k_y}(x,y) \right\rangle \tag{6.10}$$

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 (4.16). A equação (6.10) 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

$A^d_{2^{j+1}} \cdot f$ em ambas as direções. Primeiramente, $A^d_{2^{j+1}} \cdot f$ 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. 6.1.

Todas as aproximações $A^d_{2^j} \cdot f$ para $j<0$ podem ser obtidas recursivamente tendo como ponto de partida $A^d_1 \cdot f$ 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.

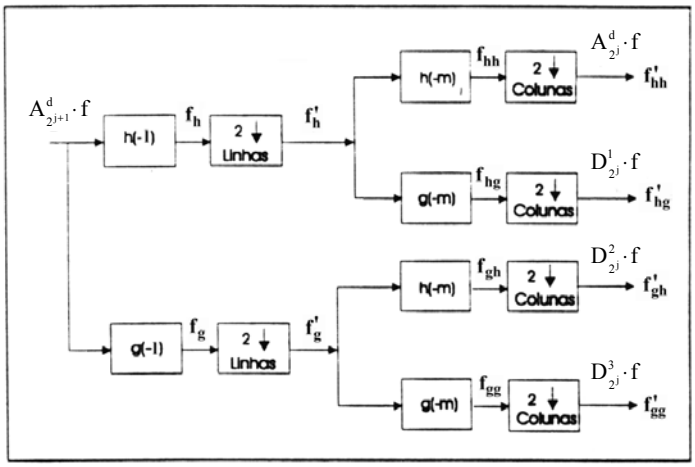


Fig. 6.1. Diagrama de blocos de um estágio de decomposição wavelet de uma imagem.

As indicações do tipo f_i , f'_i , f_{ii} , f'_{ii} na Fig. 6.1 são marcos para o exemplo prático de aplicação da transformada *wavelet* bidimensional que será dado na Seção 6.5.

6.2 - O sinal de Detalhamento 2D

Sendo $\{V^2_{2^j}\}$ 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 $W^2_{2^j}$ que, analogamente ao caso 1D, é o complemento ortogonal de $V^2_{2^j}$ em $V^2_{2^{j+1}}$.

Teorema D Estendendo o Teorema B 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 [5][10]. As três *wavelets* bidimensionais são dadas por:

$$\begin{aligned} \psi^1(x,y) &= \phi(x)\cdot\psi(y) \\ \psi^2(x,y) &= \psi(x)\cdot\phi(y) \\ \psi^3(x,y) &= \psi(x)\cdot\psi(y) \end{aligned} \tag{6.11}$$

Assim, a família $\left\{ \psi^i_{j;n_x,n_y}(x,y),\ 1 \le i \le 3 \right\}$ dada em (6.12) é uma base ortonormal para cada $W^2_{2^j}$.

$$\left\{ \begin{array}{l} \psi^1_{j;n_x,n_y}(x,y) = 2^j \psi^1\left(2^j x - n_x\,,\, 2^j y - n_y\right) \\ \psi^2_{j;n_x,n_y}(x,y) = 2^j \psi^2\left(2^j x - n_x\,,\, 2^j y - n_y\right) \\ \psi^3_{j;n_x,n_y}(x,y) = 2^j \psi^3\left(2^j x - n_x\,,\, 2^j y - n_y\right) \end{array} \right\},\ (n_x,n_y) \in Z^2 \tag{6.12}$$

Analogamente ao caso unidimensional e baseado no Teorema D, 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 $W^2_{2^j}$ [6]:

$$\begin{aligned} D^1_{2^j} \cdot f &= \left\{ \left\langle f(x,y), \psi^1_{j;n_x,n_y}(x,y) \right\rangle \quad ,(n_x,n_y) \in Z \right\} \\ D^2_{2^j} \cdot f &= \left\{ \left\langle f(x,y), \psi^2_{j;n_x,n_y}(x,y) \right\rangle \quad ,(n_x,n_y) \in Z \right\} \\ D^3_{2^j} \cdot f &= \left\{ \left\langle f(x,y), \psi^3_{j;n_x,n_y}(x,y) \right\rangle \quad ,(n_x,n_y) \in Z \right\} \end{aligned} \tag{6.13}$$

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 $W^2_{2^j} \subset V^2_{2^{j+1}}$. Pode-se então expandir $\left\{ \psi^i_{j;n_x,n_y}(x,y), 1 \le i \le 3 \right\}$ na base ortonormal de $V^2_{2^{j+1}}$:

$$\psi^i_{j;n_x,n_y}(x,y) = \sum_{k_y=-\infty}^{\infty} \sum_{k_x=-\infty}^{\infty} \left\langle \psi^i_{j;n_x,n_y}(u,v)\ ,\ \phi_{j+1;k_x,k_y}(u,v) \right\rangle \cdot \phi_{j+1;k_x,k_y}(x,y) \tag{6.14}$$

Aplicando uma mudança de variáveis, tem-se:

$$\psi^i_{j;n_x,n_y}(x,y) = \sum_{k_y=-\infty}^{\infty} \sum_{k_x=-\infty}^{\infty} \left\langle \psi^i_{-1;0,0}(u,v), \phi_{0;(k_x-2n_x),(k_y-2n_y)}(u,v) \right\rangle \cdot \phi_{j+1;k_x,k_y}(x,y) \tag{6.15}$$

Definindo o filtro discreto bidimensional de resposta impulsiva:

$$g^i(\ell,m) = \left\langle \psi^i_{-1;0,0}(u,v), \phi_{0;\ell,m}(u,v) \right\rangle \tag{6.16}$$

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 (4.16) e (4.32). Assim:

$$\begin{aligned} g^1(\ell,m) &= \left\langle \phi_{-1,0}(u)\psi_{-1,0}(v), \phi_{0,\ell}(u)\phi_{0,m}(v)\right\rangle = \left\langle \phi_{-1,0}(u), \phi_{0,\ell}(u)\right\rangle \cdot \left\langle \psi_{-1,0}(v), \phi_{0,m}(v)\right\rangle = h(\ell)\cdot g(m) \\ g^2(\ell,m) &= \left\langle \psi_{-1,0}(u)\phi_{-1,0}(v), \phi_{0,\ell}(u)\phi_{0,m}(v)\right\rangle = \left\langle \psi_{-1,0}(u), \phi_{0,\ell}(u)\right\rangle \cdot \left\langle \phi_{-1,0}(v), \phi_{0,m}(v)\right\rangle = g(\ell)\cdot h(m) \\ g^3(\ell,m) &= \left\langle \psi_{-1,0}(u)\psi_{-1,0}(v), \phi_{0,\ell}(u)\phi_{0,m}(v)\right\rangle = \left\langle \psi_{-1,0}(u), \phi_{0,\ell}(u)\right\rangle \cdot \left\langle \psi_{-1,0}(v), \phi_{0,m}(v)\right\rangle = g(\ell)\cdot g(m) \end{aligned} \quad (6.17)$$

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

$$\left\langle f(x,y)\cdot \psi^1_{j;n_x,n_y}(x,y)\right\rangle = \sum_{k_y=-\infty}^{\infty} g(k_y - 2n_y)\cdot \sum_{k_x=-\infty}^{\infty} h(k_x - 2n_x)\cdot \left\langle f(x,y)\cdot \phi_{j+1;n_x,n_y}(x,y)\right\rangle \quad (6.18)$$

$$\left\langle f(x,y)\cdot \psi^2_{j;n_x,n_y}(x,y)\right\rangle = \sum_{k_y=-\infty}^{\infty} h(k_y - 2n_y)\cdot \sum_{k_x=-\infty}^{\infty} g(k_x - 2n_x)\cdot \left\langle f(x,y)\cdot \phi_{j+1;n_x,n_y}(x,y)\right\rangle \quad (6.19)$$

$$\left\langle f(x,y)\cdot \psi^3_{j;n_x,n_y}(x,y)\right\rangle = \sum_{k_y=-\infty}^{\infty} g(k_y - 2n_y)\cdot \sum_{k_x=-\infty}^{\infty} g(k_x - 2n_x)\cdot \left\langle f(x,y)\cdot \phi_{j+1;n_x,n_y}(x,y)\right\rangle \quad (6.20)$$

As equações (6.18)-(6.20) 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. 6.1.

6.3 - 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 $-J \le j \le -1$. Um estágio completo deste algoritmo encontra-se na Fig. 6.1. 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 $A^d_1\cdot f(x,y)$ é dada pelas $3J+1$ imagens discretas:

$$\left(A^d_{2^{-J}}\cdot f\ ,\ \left\{D^1_{2^j}\cdot f\right\}_{-J\le j\le -1}, \left\{D^2_{2^j}\cdot f\right\}_{-J\le j\le -1}, \left\{D^3_{2^j}\cdot f\right\}_{-J\le j\le -1}\right) \quad (6.21)$$

A imagem $A^d_{2^{-J}} \cdot f$ é a aproximação de $A^d_1 \cdot f$ com nível de resolução 2^{-J}, correspondendo às freqüências mais baixas de *f(x,y)*. $\left\{D^i_{2^j} \cdot f\right\}_{1 \le i \le 3}$ são as imagens de detalhamento para diferentes níveis de resoluções onde: $D^1_{2^j} \cdot f$ corresponde às freqüências verticais altas (linhas horizontais na imagem), $D^2_{2^j} \cdot f$ corresponde às freqüências horizontais altas (linhas verticais na imagem) e $D^3_{2^j} \cdot f$ corresponde às freqüências altas em ambas as direções. Se a imagem original possui N *pixels*, cada imagem $A^d_{2^{-J}} \cdot f$, $D^1_{2^j} \cdot f$, $D^2_{2^j} \cdot f$ e $D^3_{2^j} \cdot f$ da representação possui 2^{2j}. *N pixels*, totalizando os mesmos *N pixels* da imagem original.

A Fig. 6.2 ilustra a representação *wavelet* ortogonal 2D para três estágios (*J=3*).

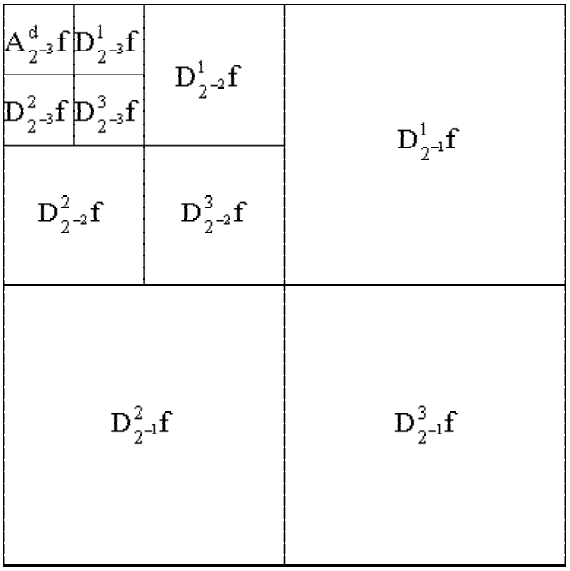


Fig. 6.2. Imagens resultantes da decomposição wavelet ortogonal bidimensional para J=3.

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

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

$$\phi_{j+1;n_x,n_y}(x,y) = \sum_{k_y=-\infty}^{\infty} \sum_{k_x=-\infty}^{\infty} \left\langle \phi_{j+1;n_x,n_y}(u,v), \phi_{j;k_x,k_y}(u,v) \right\rangle \cdot \phi_{j;k_x,k_y}(x,y) + \\ + \sum_{i=1}^{3} \sum_{k_y=-\infty}^{\infty} \sum_{k_x=-\infty}^{\infty} \left\langle \phi_{j+1;n_x,n_y}(u,v), \psi^i_{j;k_x,k_y}(u,v) \right\rangle \cdot \psi^i_{j;k_x,k_y}(x,y) \qquad (6.22)$$

Calculando o produto interno de ambos os lados de (6.22) com *f(x,y)*, tem-se:

$$\left\langle f(x,y),\phi_{j+1;n_x,n_y}(x,y)\right\rangle = \sum_{k_y=-\infty}^{\infty}\sum_{k_x=-\infty}^{\infty}\left\langle \phi_{j+1;n_x,n_y}(u,v),\phi_{j;k_x,k_y}(u,v)\right\rangle \cdot \left\langle f(x,y),\phi_{j;k_x,k_y}(x,y)\right\rangle + \\ + \sum_{i=1}^{3}\sum_{k_y=-\infty}^{\infty}\sum_{k_x=-\infty}^{\infty}\left\langle \phi_{j+1;n_x,n_y}(u,v),\psi^{i}_{j;k_x,k_y}(u,v)\right\rangle \cdot \left\langle f(x,y),\psi^{i}_{j;k_x,k_y}(x,y)\right\rangle \qquad (6.23)$$

Aplicando algumas substituições, tem-se:

$$\left\langle f(x,y)\cdot\phi_{j+1;n_x,n_y}(x,y)\right\rangle = \sum_{k_y=-\infty}^{\infty} h(n_y-2k_y)\sum_{k_x=-\infty}^{\infty} h(n_x-2k_x)\cdot\left\langle f(x,y),\phi_{j;k_x,k_y}(x,y)\right\rangle + \\ + \sum_{k_y=-\infty}^{\infty} g(n_y-2k_y)\sum_{k_x=-\infty}^{\infty} h(n_x-2k_x)\cdot\left\langle f(x,y),\psi^{1}_{j;k_x,k_y}(x,y)\right\rangle + \\ + \sum_{k_y=-\infty}^{\infty} h(n_y-2k_y)\sum_{k_x=-\infty}^{\infty} g(n_x-2k_x)\cdot\left\langle f(x,y),\psi^{2}_{j;k_x,k_y}(x,y)\right\rangle + \\ + \sum_{k_y=-\infty}^{\infty} g(n_y-2k_y)\sum_{k_x=-\infty}^{\infty} g(n_x-2k_x)\cdot\left\langle f(x,y),\psi^{3}_{j;k_x,k_y}(x,y)\right\rangle \qquad (6.24)$$

Em (6.24) nota-se que para reconstruir $A_{2^{j+1}}\cdot f$ primeiro há a inserção de uma coluna de zeros entre cada coluna de $A^{d}_{2^j}\cdot f$, $D^{1}_{2^j}\cdot f$, $D^{2}_{2^j}\cdot f$, $D^{3}_{2^j}\cdot f$ 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 (4.16) e (4.32).

Deve-se notar que cada coeficiente de $A^{d}_{2^{j+1}}\cdot f$ é formado por todos os coeficientes de $A^{d}_{2^j}\cdot f$, $D^{1}_{2^j}\cdot f$, $D^{2}_{2^j}\cdot f$ e $D^{3}_{2^j}\cdot f$. O diagrama em blocos de um estágio da reconstrução da imagem encontra-se na Fig. 6.3. A imagem $A^{d}_{1}\cdot f$ é reconstruída a partir da representação *wavelet* repetindo-se este processo para $-1 \le j \le -J$.

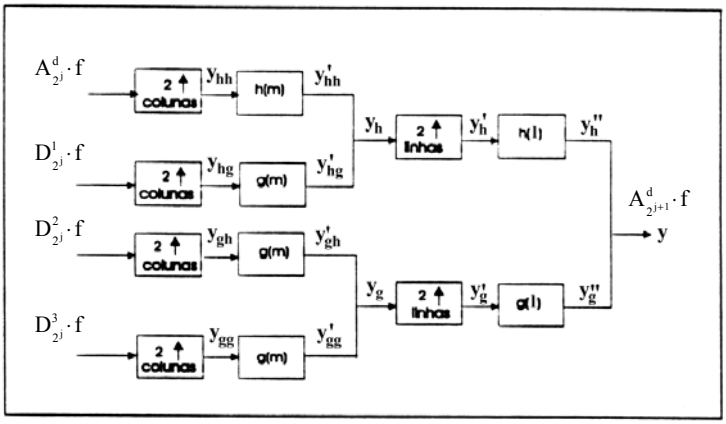


Fig. 6.3. Diagrama em blocos de um estágio da reconstrução da imagem.

6.5 – Exemplos da TWD-2D e sua inversa

Exemplo 1) A Fig. 6.4 mostra um exemplo ilustrativo de um estágio da TWD-2D aplicada a uma imagem 8x8 de um pulso gaussiano, enquanto a Fig. 6.5 mostra sua inversa [5]. As indicações do tipo f_i , f'_i , f_{ii} , f'_{ii} foram referenciadas nas Figs. 6.1 e 6.3. Os arredondamentos da implementação não foram mostrados

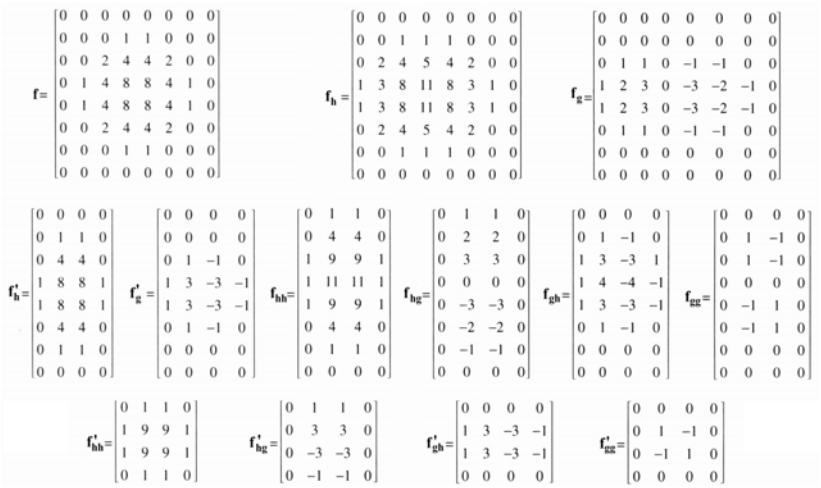


Fig. 6.4. Exemplo da aplicação da TWD bidimensional

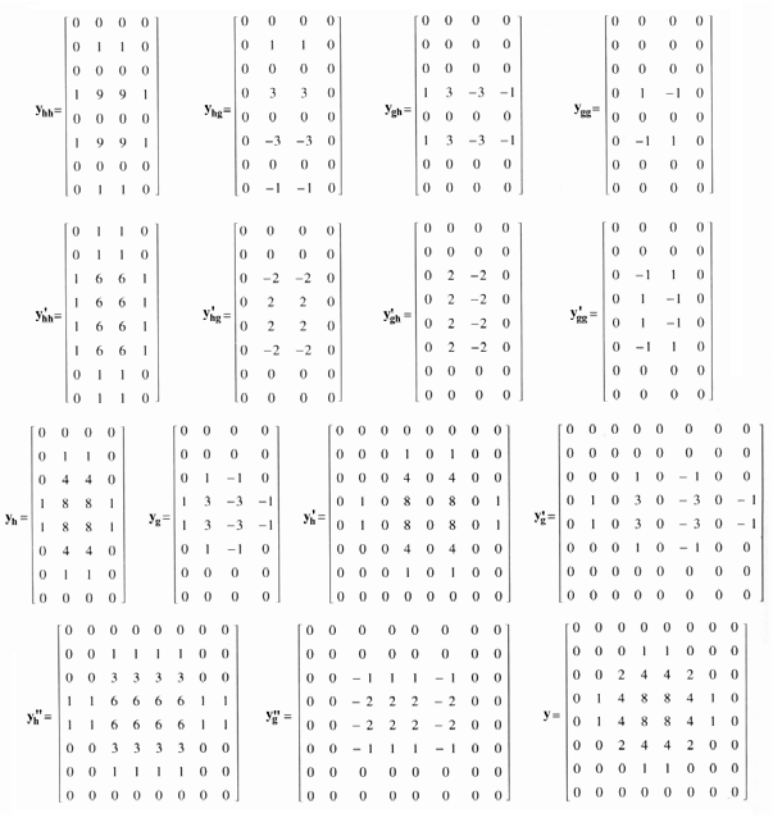


Fig. 6.5. Exemplo da aplicação da TWD bidimensional inversa.

Exemplo 2) A Fig. 6.6 mostra a imagem original *Yosemite* 248x350 e a Fig. 6.7 mostra a representação *wavelet* ortogonal bidimensional da imagem *Yosemite* para três estágios feita no aplicativo *Matlab Wavelets Toolbox* (*Matlab 6.1)*.

Fig. 6.6. Imagem original *Yosemite* 248x350

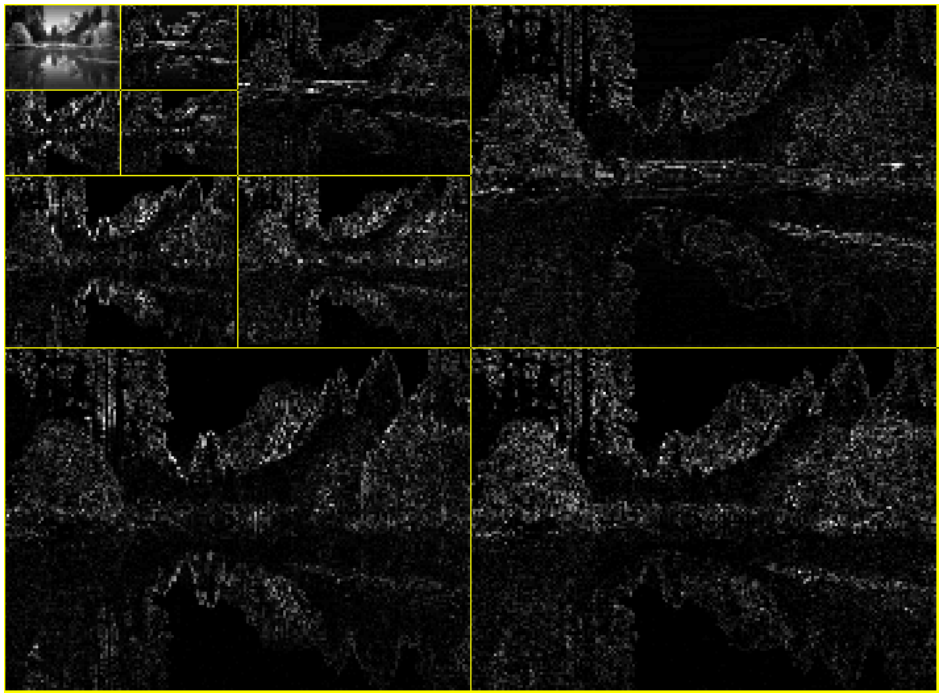


Fig. 6.7. Representação wavelet ortogonal 2D para três estágios (J=3).

6.6 – Comentários

Foram apresentadas neste capítulo a decomposição e síntese da TWD bidimensional ortogonal, bem como os sinais PB e de detalhes bidimensionais resultantes do processo de decomposição. O capítulo seguinte finalizará a apresentação da teoria *wavelet* abordando as características desejáveis da TWD para a compressão de imagens.

Capítulo 7

Regularidade, Seleção das *Wavelets* e SPIHT

Genericamente falando, as mais diversas técnicas de compressão de imagens surgem da anulação ou da quantização "grosseira" dos coeficientes de alta freqüência. Nesse contexto, a regularidade (suavidade) da TW está diretamente relacionada com a fidelidade na representação da função. Isso ocorre uma vez que, se a *wavelet* não for regular (apresentando, por exemplo, descontinuidades ou quebras), serão gerados mais componentes de alta freqüência, o que em um sistema de compressão acarretaria erros e perda de qualidade visual.

Este capítulo tem por objetivo apresentar as características desejáveis para a TW na compressão de imagens (alvo desta pesquisa), 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. Este algoritmo é apresentado por ser o núcleo do Sistema-Base 1 adotado para comparação com o codificador proposto.

7.1 – *Wavelets* de Daubechies

Daubechies [16] construiu 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:

$$\int_{-\infty}^{\infty} x^n \psi(x)|_N \, dx = 0 \qquad n = 0,1,\ldots N \tag{7.1}$$

e seu número de derivadas contínuas é ≅ N/5. Isso descreve um grupo de funções bem comportadas, também chamadas de *wavelets regulares*.

Em [16] pode ser encontrado o algoritmo de Daubechies para a construção de filtros FIR *h(n)* que gerem as *wavelets* $\psi(x)$ regulares, ortogonais e de suporte compacto. Este algoritmo consiste na fatoração de um polinômio função de *N* (onde *2N* é o comprimento do filtro *h(k)*) e na aplicação das fórmulas (4.46), (4.21) e (4.47), não sendo apresentado aqui dada a sua extensão e por fugir ao escopo da tese.

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. 7.1. Curiosamente, $\psi(x)\big|_{N=1}$ é a *wavelet*-básica da transformada de Haar. A Tabela VII.1 mostra os coeficientes dos filtros *h(n)* das *wavelets*-básicas com $N = 3, 5, 7$ e 9.

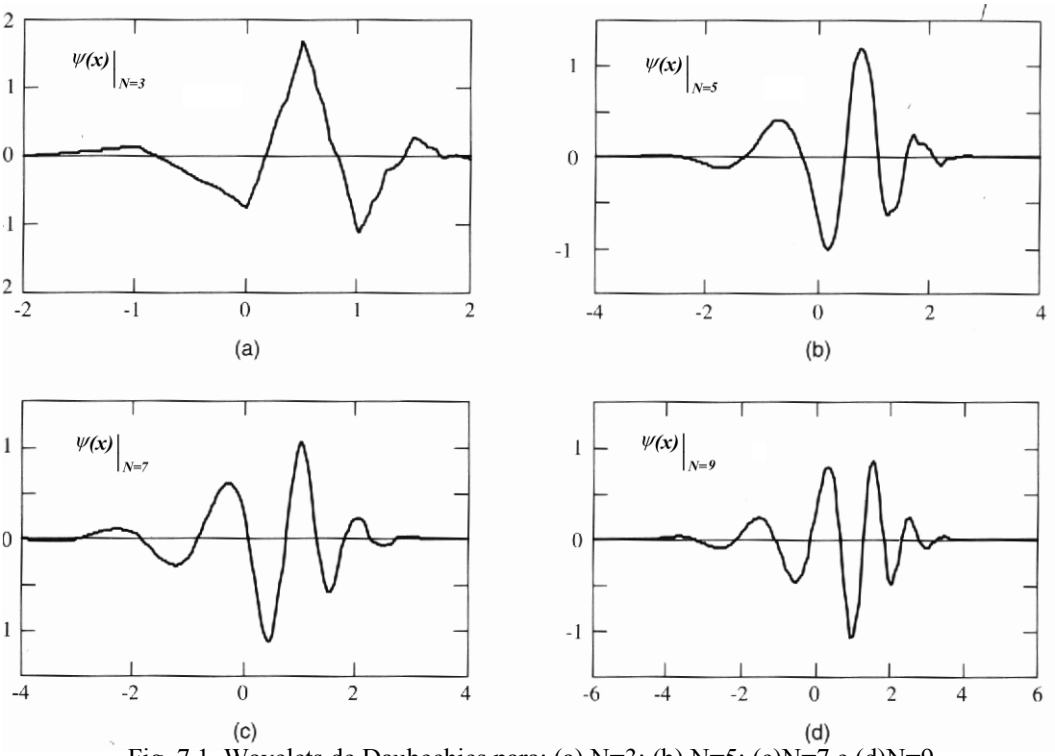


Fig. 7.1. Wavelets de Daubechies para: (a) N=3; (b) N=5; (c)N=7 e (d)N=9.

TABELA VII.1 – COEFICIENTES DOS FILTROS *H(N)* DAS *WAVELETS*-BÁSICAS COM $N = 3, 5, 7$ E 9.

N	**Coefs**										
3	$\mathbf{h_0}$-$\mathbf{h_5}$	0,3327	0,8069	0,4599	-0,1350	-0,0854	0,0352				
5	$\mathbf{h_0}$-$\mathbf{h_9}$	0,1601	0,6083	0,7243	0,1384	-0,2423	-0,0322	0,0776	-0,0062	-0,0126	0,0033
7	$\mathbf{h_0}$-$\mathbf{h_6}$	0,0779	0,3965	0,7291	0,4698	-0,1439	0,2240	0,0713			
	$\mathbf{h_7}$-$\mathbf{h_{13}}$	0,0806	-0,0380	-0,0166	0,126	0,0004	-0,0018	0,0004			
9	$\mathbf{h_0}$-$\mathbf{h_8}$	0,0381	0,2438	0,6048	0,6573	0,1332	-0,2933	-0,0968	0,1485	0,0307	
	$\mathbf{h_9}$-$\mathbf{h_{17}}$	-0,0676	0,0003	0,0224	-0,0047	-0,0043	0,0018	0,0002	-0,0003	0,0000	

7.2 – Seleção das *Wavelets*.

A aplicação em processamento de imagens requer que a TW e, portanto seus filtros de implementação, apresentem certas características [5][6][9]:

- 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.
- 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 possíveis de serem representadas com fidelidade.

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* [15] 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 [15].

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 têm sido uma área ativa de pesquisa. Vários autores têm catalogado filtros e suas *wavelets* biortogonais correspondentes.

7.3 – Implementação das *Wavelets* Biortogonais

Em [15][32] 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 [15][ref. 12 de 15] provou-se que uma alta regularidade pode ser obtida, tanto para ψ quanto para $\tilde{\psi}$, desde que se selecione os filtros suficientemente longos. Em particular, se as funções ψ e $\tilde{\psi}$ forem, respectivamente, diferenciáveis $(p-1)$ e $(\tilde{p}-1)$ vezes no tempo, então suas respostas em freqüências (*H(w)* e $\tilde{H}(w)$) são divisíveis por $(1+e^{-jw})^{p}$ e $(1+e^{-jw})^{\tilde{p}}$, respectivamente. Logo, os filtros (*h(n)* e $\tilde{h}(n)$) terão um cumprimento maior que p e $\tilde{p}$, respectivamente [33].

Outro aspecto é que a divisibilidade de $\tilde{H}(w)$ por $(1+e^{-jw})^{\tilde{p}}$ significa que ψ pode ter $\tilde{p}$ momentos nulos consecutivos [9], ou seja,

$$\int x^{l}\psi(x)dx = 0, \qquad para\ l = 0,1,\ldots..,\ \tilde{p}\ -\ 1 \tag{7.2}$$

Prova-se utilizando-se as séries de Taylor [9], 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 [15] 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 [15]:

$$H(w)\tilde{H}(w) + H(w+\pi)\tilde{H}(w+\pi) = 2 \tag{7.3}$$

Combinando-se as expressões (7.2) e (7.3) e levando-se em conta a imposição de divisibilidade de *H(w)* e $\tilde{H}(w)$ por $(1+e^{-jw})^{p}$ e $(1+e^{-jw})^{\tilde{p}}$, chega-se através de uma prova bastante extensa à [15]:

$$H(w)\tilde{H}(w) = cos^{2l}\left(w/2\right)\cdot\left[\sum_{q=0}^{l-1}\binom{l-1+q}{q} sin^{2q}\left(w/2\right) + sin^{2l}\left(w/2\right) R(w)\right] \tag{7.4}$$

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 (7.4) 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 [33]. Na Tabela VII.2 são apresentados os coeficientes h_n e $\tilde{h}_n$ dos filtros passa-baixas destes três exemplos e nas Figs. 7.2, 7.3, 7.4 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 [9]:

$$\tilde{g}(n) = (-1)^{n-1} \cdot h^*(1-n) \tag{7.5}$$

$$g(n) = (-1)^{n-1} \cdot \tilde{h}^*(1-n) \tag{7.6}$$

Tabela VII.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]	$2^{-1/2}.\ h_n$	45/64	19/64	-1/8	3/64	3/128
	$2^{-1/2}.\ \tilde{h}_n$	1/2	1/4	0	0	0
Filtros variante *spline* com comprimentos semelhantes $p=4$, $\tilde{p}=4$ e $l=4$ [9-7]	$2^{-1/2}.\ h_n$	0,602949	0,266864	-0,078223	-0,016864	0,026749
	$2^{-1/2}.\ \tilde{h}_n$	0,557543	0,295636	-0,028772	-0,045636	0
Filtros variante *spline* com comprimentos muito semelhantes $p=2$, $\tilde{p}=2$ e $l=2$ [5-7]	$2^{-1/2}.\ h_n$	0,6	0,25	-0,05	0	0
	$2^{-1/2}.\ \tilde{h}_n$	17/28	73/280	-3/56	-3/280	0

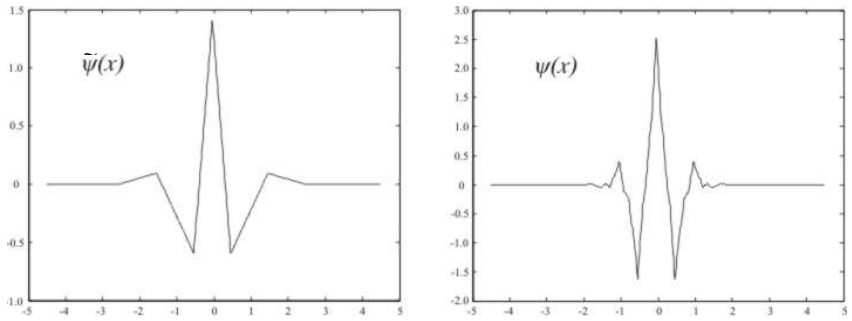


Fig. 7.2 – Wavelet biortogonal *spline* [9-3] e sua dual

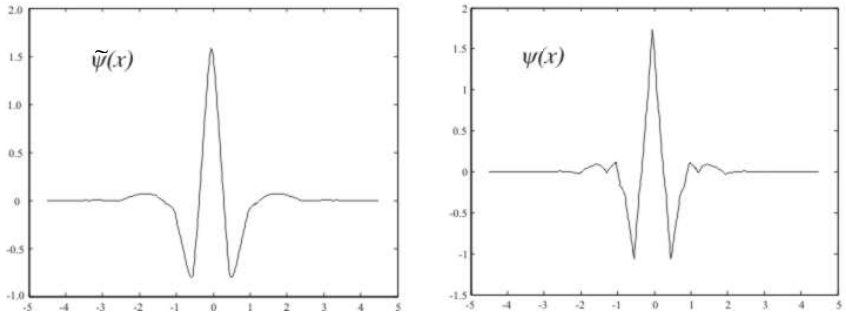


Fig. 7.3 – Wavelet biortogonal *spline* [9-7] e sua dual

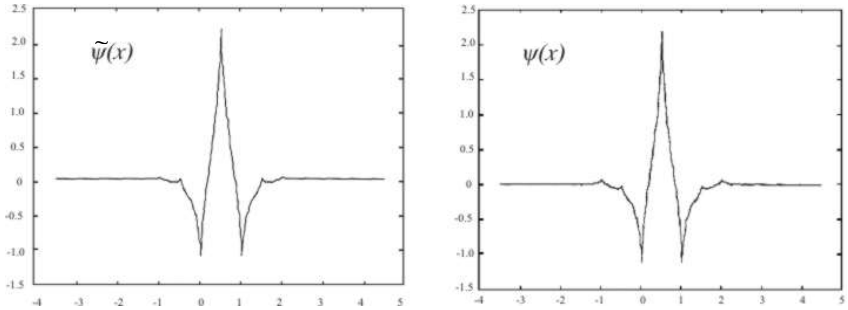


Fig. 7.4Wavelet biortogonal variante *spline* [5-7] e sua dual

7.3.a – *Wavelets Spline* Biortogonais Cohen-Daubechies-Feauveau

Fazendo *R(w) =0* em (7.4), e se a resposta do filtro passa-baixas dual for definida por:

$$\tilde{H}(w) = cos^{\tilde{p}}\left(w/2\right) \cdot e^{-j\varepsilon w}, \text{ onde } \begin{cases} \varepsilon = 0, \text{ se } \tilde{p} \text{ é par} \\ \varepsilon = 1, \text{ se } \tilde{p} \text{ é ímpar} \end{cases} \tag{7.7}$$

obtém-se a família de filtros conhecida como filtros *spline* [15], 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:

$$H(w) = cos^{2l-\tilde{p}}\left(w/2\right) e^{\frac{jw\varepsilon}{2}} \left[\sum_{q=0}^{l-1} \binom{l-1+q}{q} sin^{2q}\left(w/2\right) \right] \tag{7.8}$$

Na Tabela VII.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 cumprimento ímpar. Essa é uma característica dos filtros *spline* [33].

7.3.b – Variante de *Wavelets Spline* Biortogonais

Para se construir essa família de filtros faz-se *R(w) =0* em (7.4) e fatora-se o lado direito da expressão quebrando o polinômio de grau *l-1* em $sin\left(w/2\right)$, 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 $\tilde{H}$ [33].

O exemplo na Tabela VII.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. 7.5 ilustra a *wavelet* biortogonal spline [9,7] 2D adotada nesta pesquisa.

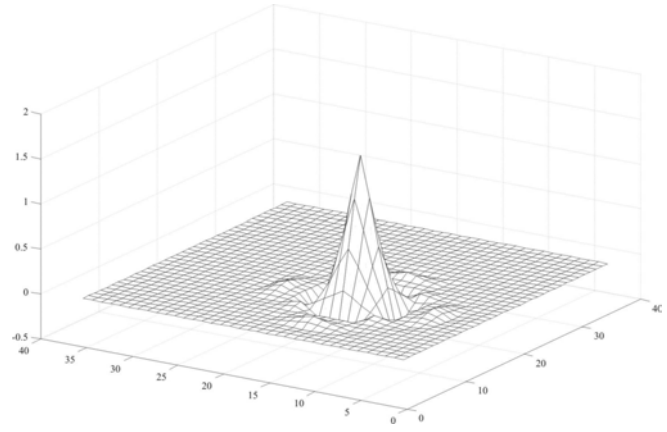


Fig. 7.5. Wavelet biortogonal *spline* [9,7] 2D

7.3.c –*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 (7.4) estão muito próximos e conseqüentemente, muito próximos ao caso ortonormal. Um desses casos é o filtro piramidal laplaciano proposto em [34], para o qual $l=p=2$ e

$$R(w) = 48\cos\left(\frac{w}{2}\right)\Big/175 \tag{7.9}$$

Os coeficientes desse filtro podem ser consultados na Tabela VII.2. Os dois filtros biortogonais desse exemplo são ambos muito próximos de um filtro *wavelet* ortonormal de comprimento 6 apresentado em [35] 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 [33].

7.4 – SPIHT (*Set Partitioning in Hierarchical Trees*)

O Codificador *Wavelet* "Puro" que foi usado como um dos Sistemas-Base para Comparação com o Codificador Híbrido proposto nesta pesquisa tem como núcleo a TWD e um algoritmo de compressão *lossy* conhecido como SPIHT (*Set Partitioning in Hierarchical Trees*). Nos testes realizados nesta pesquisa (Capítulos 12 à 14), este Codificador *Wavelet* "Puro" será referenciado simplesmente como "TWD+SPIHT". Neste contexto, esta seção destina-se a fornecer uma breve apresentação do algoritmo SPIHT em si, bem como análises acerca dessa escolha.

O algoritmo SPIHT foi introduzido por Said e Pearlman [36][37][38] e se apresenta como uma versão bastante refinada do EZW tradicional [39][40]. 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 [41] 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 subimagens 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 *threshold*, sua descendência tende fortemente a apresentar magnitudes menores que esse mesmo *threshold.* 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 subárvore 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 melhor agrupados em *subconjuntos maiores* [42].

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 [41].

Os *thresholds* 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.

Notação e Princípio de Funcionamento

Os coeficientes-*wavelet* são divididos em árvores originárias à partir da resolução mais baixa (banda I na Fig. 7.6). Estes coeficientes são agrupados em *arrays* 2x2 que, exceto para a banda I, são filhos de um coeficiente numa subbanda de resolução mais baixa. Os coeficientes na subbanda I são também agrupados em *arrays* 2x2, entretanto, o coeficiente-*wavelet* do canto superior esquerdo de cada *array* dessa subbanda não é tratado como um nó-raiz, não sendo vinculado a ele qualquer descendência. Essa estrutura pode ser observada na Fig 7.6 para uma TWD de 2 níveis [42].

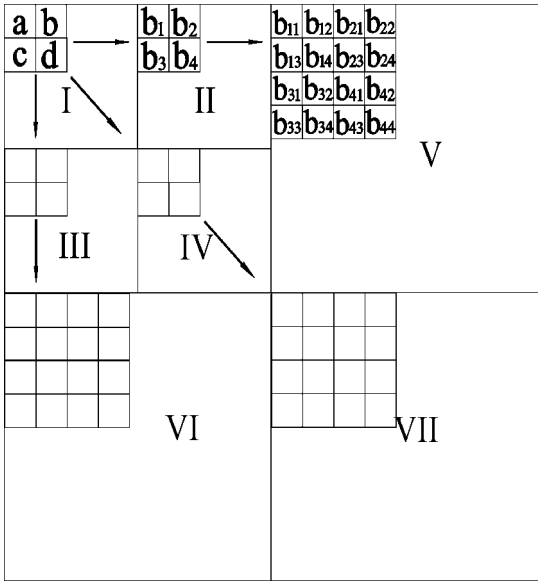


Fig. 7.6 – Estrutura de dados utilizada pelo SPIHT[43]

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. 7.6 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 7.6 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 7.6, 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 7.6 o conjunto $\mathcal{L}(1,2)$ consiste das coordenadas dos coeficientes $b_{11}, ..., b_{14}, ..., b_{44}$. Portanto:

$$\mathcal{L}(i,j) = \mathcal{D}(i,j) - \mathcal{O}(i,j) \tag{7.10}$$

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 *threshold*. Esses *thresholds* 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;
- 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 *threshold*, como será visto no exemplo.

Acompanhamento

O algoritmo tem início com a determinação de valor do *threshold* inicial (T_0), dado por:

$$T_0 = 2^n \tag{7.11}$$

onde, sendo c_{max} a magnitude máxima dos coeficientes-*wavelet* a codificar, n é dado por:

$$n = \lfloor \log_2 c_{\max} \rfloor \tag{7.12}$$

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 os componentes de LIP e a seguir os de 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 de 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 LSP.

Após o processamento de todos os componentes de LIP, os conjuntos componentes de 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, pelo menos um componente 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 subimagem. 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 este passo. 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 esse passo. 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 $|c_{ij}|$. 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.

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*.

26	6	13	10
-7	7	6	4
4	-4	4	-3
2	-2	-2	0

Passo 1) $c_{max} = 26$. Logo, n = 4. $T_0 = 16$. A inicialização das listas, portanto, é dada por:

LIP: $\{ (1,1) \rightarrow 26; (1,2) \rightarrow 6; (2,1) \rightarrow -7; (2,2) \rightarrow 7 \}$

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 16) e '*0*' (significativo positivo) e suas coordenadas são movidas para a LSP. Os próximos 3 coeficientes de LIP são insignificantes com relação a 16, 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 13, 10, 4 e 6 (segundo a ordem da varredura), observa-se que nenhum deles é significativo com relação a 16. Assim, para esse conjunto $(1,2)\mathcal{D}$, transmite-se um '*0*'. Idem para os conjuntos $(2,1)\mathcal{D}$ e $(2,2)\mathcal{D}$. Logo, para cada um deles também transmite-se um '*0*'.

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: $\{ (1,2) \rightarrow 6; (2,1) \rightarrow -7; (2,2) \rightarrow 7 \}$

LIS: $\{ (1,2)\mathcal{D}; (2,1)\mathcal{D}; (2,2)\mathcal{D} \}$

LSP: $\{(1,1) \rightarrow 26\}$

Bitstream: 10 000 000 = 8 bits

Passo 2) Feito o decremento de 1 em *n*, n = 3. $T = 8$.

Novamente, iniciando pelo exame da LIP. Há 3 coeficientes na LIP, sendo todos insignificantes para este novo *threshold*. Logo, são transmitidos um zero referente a cada um deles e eles permanecem em LIP.

Segue-se o exame de LIS. Olhando para a descendência de (1,2), observa-se que 13 e 10 são significativos com relação a 8. Assim, para este conjunto $(1,2)\mathcal{D}$, transmite-se um '*1*'e abre-se a análise de sua descendência direta (os filhos 13, 10, 4 e 6). O valor 13 é significativo positivo, logo, transmite-se um '*1*' seguido de um '*0*'. O valor 10 também é significativo positivo, logo, transmite-se um '*1*' seguido de um '*0*' novamente. O valor 4 é insignificante, logo, transmite-se um '*0*', o mesmo ocorrendo para o valor 6. As coordenadas de 13 e 10 são então movidas para LSP e as coordenadas de 4 e 6 são movidas para a LIP. Como $(1,2)\mathcal{L}$ é vazio, o conjunto $(1,2)\mathcal{D}$ é então retirado de LIS (em outras palavras, uma vez que não há netos para (1,2)).

Os conjuntos $(2,1)\mathcal{D}$ e$(2,2)\mathcal{D}$ não apresentam filhos significativos, logo, para cada um deles transmite-se um '*0*'. A seguir, são examinados os coeficientes de LSP *do passo anterior*, que consiste no coeficiente 26. Portanto, o refinamento consiste em mandar o (*n+1*) bit menos significativo de 26, que é '*1*'.

A situação das listas no fim desse segundo passo é:

LIP: $\{\ (1,2) \to 6; (2,1) \to -7; (2,2) \to 7; (2,4) \to 4; (2,3) \to 6\ \}$

LIS: $\{\ (2,1)\mathcal{D};\ (2,2)\mathcal{D}\ \}$

LSP: $\{(1,1) \to 26; (1,3) \to 13; (1,4) \to 10;\}$

Bitstream: 000 1101000 0 0 1= 13 bits adicionais.

Passo 3) n = 2. $T = 4$.

Seguindo a orientação dos procedimentos, pode-se verificar que a situação das listas e o *bitstream* transmitido no fim deste terceiro passo são:

LIP: $\{\ (4,1) \to 2; (4,2) \to -2; (3,4) \to -3; (4,3) \to -2; (4,4) \to 0\ \}$

LIS: $\{\ \}$

LSP: $\{(1,1) \to 26; (1,3) \to 13; (1,4) \to 10; (1,2) \to 6; (2,1) \to -7; (2,2) \to 7;$

$(2,4) \to 4;\ (2,3) \to 6;\ (3,1) \to 4; (3,2) \to -4; (3,3) \to 4\}$

Bitstream: 1011101010 1100011 110000 010= 26 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 [39] 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 TWD (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*.

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 $T_0=16$. 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:

24	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* 000 1101000 0 0 1 recebido é examinado. Como só há 3 entradas em LIP e os 3 primeiros bits (000) são *0*, estes componentes são ainda insignificantes. Os próximos 9 bits (1101000 0 0) dão informação a respeito de LIS. Percebe-se que o conjunto de raiz (1,2) é significativo (1), logo, é aberta a sua primeira descendência: 101000 (significativo positivo, significativo positivo, insignificante e insignificante; segundo a ordem de varredura, também conhecida do decodificador). Logo, os significativos vão para a LSP e

os insignificantes para LIP. Esses coeficientes significativos são aproximados para a reconstrução por 1.5 x 2^3 =12. O conjunto $(1,2)\mathcal{D}$ é também removido de LIS.

Os próximos bits (0 0) indicam que os dois conjuntos que sobraram em LIS são insignificantes e o bit final corresponde ao refinamento de (1,1). Logo, a reconstrução de (1,1) é 24+8/2 = 28.

28 0	0 0	12 0	12 0
0 0	0 0	0 0	0 0

LIP: $\{(1,2);(2,1);(2,2);(2,4);(2,3)\}$

LIS: $\{(2,1)\mathcal{D};(2,2)\mathcal{D}\}$

LSP: $\{(1,1);(1,3);(1,4);\}$

Para os demais passos, basta dar continuidade ao processo, refinando ainda mais a imagem. Pode-se perceber a natureza da transmissão progressiva apresentada também pelo SPIHT. Estudos mais recentes com relação a melhorias aplicadas sobre o SPIHT podem ainda ser encontrados em [43]-[46].

7.4 –Comentários

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. Por fim, é apresentada a TWD *spline* biortogonal 9-7, adotada por este trabalho e suas características.

Este capítulo encerra a abordagem da teoria de transformada *wavelet*, na qual foram fundamentados todos os conceitos necessários à apresentação no Capitulo 12 de: 1) a parcela *wavelet* do Codificador Híbrido Fractal-*Wavelet* proposto neste trabalho e 2) o Sistema-Base 1 (Codificador *Wavelet* "Puro", referenciado por "TWD+SPIHT") de comparação.

Capítulo 8

Bases da Compressão Fractal de Imagens

A codificação fractal de imagens consiste, na prática, em representar os blocos da imagem através de coeficientes de transformações contrativas, explorando o conceito de auto-similaridade. Assim, nesse tipo de codificação, ao invés de armazenar/transmitir os blocos da imagem como uma coleção de *pixels*, somente são enviados/armazenados os coeficientes dessas transformações. Esse princípio de funcionamento permite obter altas taxas de compressão mantendo ótima qualidade visual; características que têm despertado recentemente o interesse de diversos pesquisadores, dada a explosão das aplicações digitais de imagem.

Neste capítulo, apresentam-se os princípios básicos da compressão fractal, uma vez que esta pesquisa apresenta um Codificador Híbrido Fractal-*Wavelet*. São introduzido os conceitos de transformação afim, iteração e atrator, sendo esse último o mais importante. Trata-se de uma abordagem inicial com caráter ilustrativo, para num segundo momento (Capítulos 9 e 10) apresentar a abordagem matemática, da qual derivarão estes fundamentos. Esses tópicos serão importantes para compreender porque e como a compressão fractal de imagens funciona. O Capítulo 11 apresentará métodos para acelerar a compressão fractal, uma vez que o tempo de processamento exaustivo é a causa da hesitação na adoção de aplicações dessa técnica. O Capítulo 11 também apresentará os dois métodos bases de comparação adotados por este trabalho.

8.1 – Introdução ao Princípio Fractal

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, por hipótese, plana e suave; conceito simples que gera bons resultados para a maioria das aplicações práticas. Assim, ampliando-se a foto de uma montanha, via modelo clássico, ver-se-á apenas uma grande área verde plana e uniforme.

O termo *fractal* foi primeiramente introduzido por Benoit Mandelbrot em 1983 [47]. A propriedade-chave que caracteriza os fractais e os diferencia das demais técnicas é a auto-similaridade, isto é, os fractais apresentam a mesma complexidade de detalhamento independente da escala em que são observados. Ampliando-se a foto da montanha, por exemplo, veríamos o mesmo nível de detalhamento que o da imagem sem ampliação, assemelhando-se mais fielmente ao mundo real. O que imediatamente nos remete a um processo iterativo de geração dos fractais.

Barnsley and Sloan [48] foram os primeiros a reconhecer o potencial da aplicação da teoria de IFS (*Iterated Funcion Systems*) ao problema da compressão de imagens e patentearam sua idéia em 1990 e 1991 [49][50]. Jacquin em 1992 [51] introduziu um método de codificação fractal que utilizava um sistema de blocos-domínio e blocos-range, base da maioria dos codificadores fractais atuais. A aproximação colocada a seguir não particiona a imagem em blocos, como é feito nos codificadores atuais, mas foi escolhida como introdução por constituir a inspiração destes.

Princípio Fractal

Considere-se inicialmente uma folha branca de papel com um sistema de coordenadas (x,y), no qual escolhe-se um ponto arbitrário A. Selecionando aleatoriamente uma das transformações afins abaixo e aplicando-a nesse ponto, obtém-se as coordenadas de um novo ponto B. Marca-se o ponto B no papel.

$$\begin{aligned}
&\omega_1 : (x, y) \rightarrow (0.85x+0.04y, -0.04x+0.85y+40),\\
&\omega_2 : (x, y) \rightarrow (0.20x-0.26y, 0.23x+0.22y+40),\\
&\omega_3 : (x, y) \rightarrow (-0.15x+0.28y, 0.26x+0.24y+11),\\
&\omega_4 : (x, y) \rightarrow (0, 0.16y),
\end{aligned} \tag{8.1}$$

Selecionando-se uma outra transformação para o ponto B, gera-se um ponto C em outras coordenadas e assim, por diante. A união dos pontos B, C, ..., gerados pelas transformações no plano, produz uma nova imagem. Se a essa nova imagem, reaplicar-se as mesmas transformações, obter-se-á uma terceira imagem. Repetindo-se esse processo indefinidamente e sendo pacientes e persistentes o suficiente, notar-se-á que a imagem produzida pelo processo converge para uma determinada imagem específica, denominada "atrator" [52]. No caso das transformações listadas em (8.1), uma folha de samambaia surgirá no papel (Fig. 8.1a).

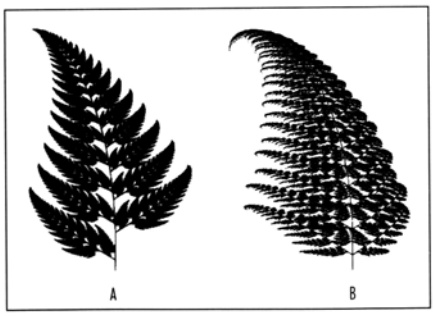


Fig. 8.1 – Folhas de samambaia: a) 2D; b) 3D. [52]

Pode-se definir *fractal* como sendo uma imagem ou figura que pode ser completamente descrita com suas texturas infinitamente detalhadas por um algoritmo matemático. 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 .

Transformações afins planares podem ser generalizadas para o espaço 3D. Nesse caso, cada transformação é caracterizada não por 6, mas por 12 coeficientes. No caso da folha de samambaia apresentada na Fig.8.1b:

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

Uma analogia simples foi feita por Fisher [3] 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. 8.2.

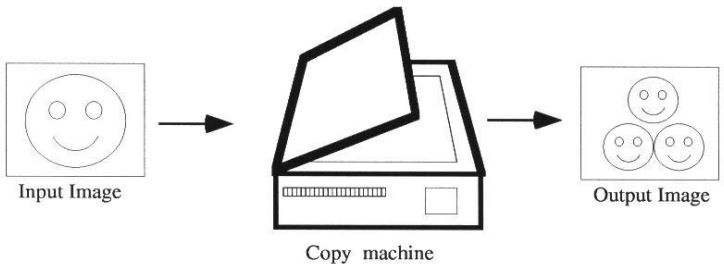


Fig. 8.2 – Exemplo de Fotocopiadora de Fisher. [3]

A idéia básica consiste em passar essa cópia de saída pela copiadora, num processo recursivo. Após várias iterações desse processo, percebe-se na Fig. 8.3 que *mesmo com diferentes imagens de entrada*, todas as cópias de saída parecem convergir para a *mesma imagem final*, mostrada na Fig. 8.3c. 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 na medida em que aumentam as iterações. Logo, a imagem inicial não afeta o atrator final; de fato somente a posição e orientação das cópias (*transformações afins*) determinam como a imagem final se parecerá.

As transformações 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 do que a distância entre os pontos correspondentes na imagem de entrada [3].

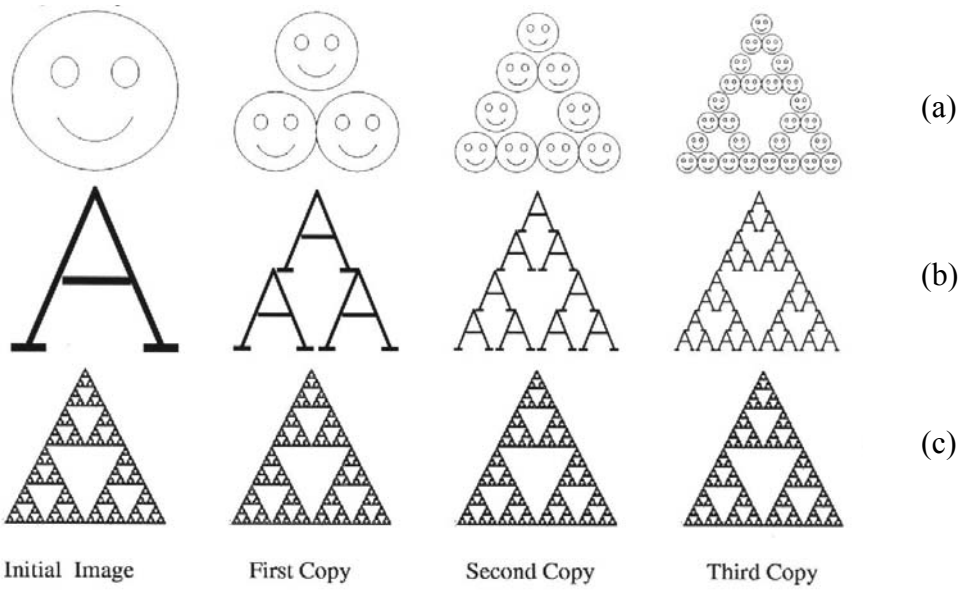


Fig. 8.3 – Primeiras 3 cópias geradas pela fotocopiadora para 3 imagens diferentes. [3]

Na prática, transformações na forma (8.2) permitem gerar uma grande variedade de interessantes atratores. Essas transformações podem torcer, deslizar, rotacionar, escalar ou deslocar a imagem de entrada, dependendo dos valores dos coeficientes.

$$\omega_i \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \end{bmatrix} \tag{8.2}$$

A Fig. 8.4 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 $\vdash$, que ajuda a visualizar as transformações [3].

O primeiro exemplo da Fig. 8.4 mostra as transformações usadas na fotocopiadora da Fig. 8.2. 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. 8.1. Os dois primeiros exemplos da Fig. 8.4 são constituídos de três transformações, enquanto o último, como visto, utiliza quatro.

A característica comum a todos os atratores frutos desse tipo de transformação é que, ao se ampliar uma de suas regiões com detalhes, ver-se-á uma cópia do atrator. Noutras palavras, cada atrator é formado por cópias reduzidas e transformadas de si mesmo. Trata-se da já citada "auto-similaridade". Esse método de gerar fractais foi criado por John Hutchinson [53].

Identificando as auto-similaridades fractais das imagens, os esforços de pesquisadores como Barnsley levaram a um novo campo científico: a compressão e representação fractal de imagens.

Barnsley [54] sugeriu que armazenar 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 folha de samambaia Fig. 8.1 armazenar-se-ía originalmente 65.536 bits (256x256, 1bpp). Com a tecnologia fractal só seriam necessários 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 é a determinação desses coeficientes e o desenvolvimento de algoritmos rápidos para a realização desse processo.

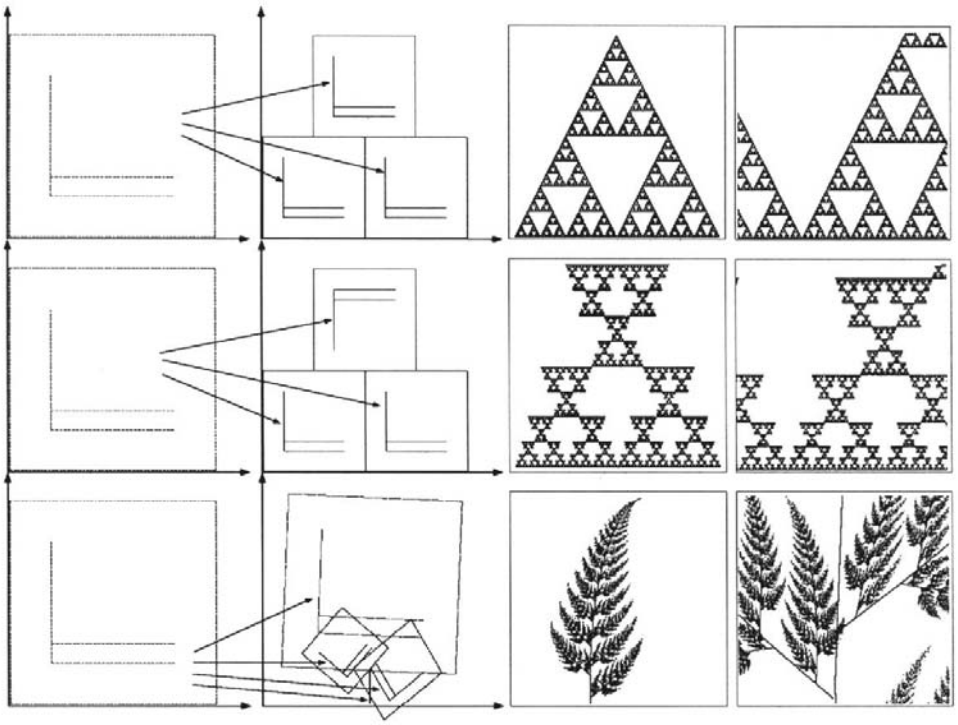


Fig. 8.4 – Transformações Afins, Atratores e Ampliação no Atrator. [3]

8.2 – Comentários

Neste capítulo, foram apresentados de forma ilustrativa os princípios básicos da compressão fractal que inspiraram os codificadores fractais atuais. Os capítulos seguintes (Capítulos 9 e 10) dão seqüência a essa apresentação, inserindo a abordagem matemática pertinente. O Capítulo 11 apresentará métodos para acelerar a compressão fractal, uma vez que o tempo de processamento exaustivo dos codificadores fractais práticos é a causa da hesitação na adoção de aplicações dessa técnica. Reside nesse aspecto o alvo desta pesquisa: a aceleração fractal através da prévia decomposição *wavelet*, explorando-se as características da última.

Capítulo 9

Espaços Métricos Completos e Contratividade

A topologia dos espaços métricos é uma parte da Matemática que estuda as conformações e características dos espaços e objetos. Em topologia de espaços métricos, o termo ($\mathcal{H}(\boldsymbol{X})$, h) define o espaço métrico onde os fractais existem. O termo h diz respeito à métrica que rege esse espaço, chamada "métrica de *Hausdorff*".

Neste capítulo, é apresentada a abordagem matemática relativa aos conceitos fractais ilustrados no Capítulo 8. Destaque especial deve ser dado aos dois teoremas principais: o teorema da colagem e o teorema do mapeamento contrativo. O primeiro deles permite que, dado o atrator, possa-se determinar o mapeamento contrativo. Portanto, pode-se determinar os coeficientes das transformações afins a serem enviados, atuando na etapa de codificação fractal. O teorema do mapeamento contrativo por sua vez garante a convergência dos mapeamentos contrativos definidos em espaços métricos completos, tais como o espaço ($\mathcal{H}(\boldsymbol{X})$, h). Noutras palavras, ele garante que o processo fractal converge para um atrator, atuando na etapa de decodificação fractal.

Também é apresentado o método de geração de fractais conhecido como IFS (*Iterated Function System* ou Sistema de Funções Iterativas), que é a inspiração dos sistemas de codificação fractais atuais. As provas matemáticas individuais dos teoremas deste capítulo serão apresentadas nos apêndices, de forma a não prejudicar a fluência da exposição dos conceitos. Maiores detalhes podem ser encontrados também em [55]. A notação {.} será usada para designar uma seqüência ou um conjunto.

9.1 - Definições Matemáticas: Espaços Métricos Completos

A primeira etapa para a definição dos espaços métricos completos, tais como o espaço ($\mathcal{H}(\boldsymbol{X})$, h) onde os fractais existem, é a definição do que é um espaço métrico e de métrica [56].

Métricas e Espaços métricos

Basicamente, espaço métrico é um espaço qualquer que tenha uma função associada que mede a distância entre os seus elementos. Essa distância precisa ser uma métrica, portanto, precisa obedecer às condições 1-4 dadas a seguir [3].

Definição 1: "Seja um espaço $\boldsymbol{X}$ e uma aplicação (função) $d: \boldsymbol{X}\times\boldsymbol{X} \rightarrow \mathfrak{R}$, o par ($\boldsymbol{X}$,$d$) é um espaço métrico se [1] [52] [56]:

1) $0 < d(x,y) < \infty \Leftrightarrow x \neq y, \ \forall x, y \in \boldsymbol{X}$;

2) $d(x,x)=0 \ \forall x \in \boldsymbol{X}$;

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)". (9.1)

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

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

$$d_{L_p}(x,y) = \left(\sum_{i=1}^{n} \left| x_i - y_i \right|^p \right)^{1/p}, \text{ para } x, y \in \boldsymbol{\Re^n} \tag{9.2}$$

$$d_{L_p}(f,g) = \left\| f(x), g(x) \right\|_p = \left(\int_X \left| f(x) - g(x) \right|^p \, dx \right)^{1/p}, \text{ para } f, g \in \{ h \mid h: \boldsymbol{\Re} \rightarrow \boldsymbol{\Re} \}. \tag{9.3}$$

Dentro dessas métricas são incluídas a métrica do módulo ($p=1$), a métrica euclidiana ou *rms* ($p=2$) e a métrica do máximo ($p \rightarrow \infty$). Uma vez definido o conceito de espaço métrico, a próxima etapa é a definição do conceito de completitude de um espaço métrico. Esta definição é feita com o auxílio de seqüências de Cauchy [56], que serão apresentadas no próximo item.

Seqüências de Cauchy

Definição 2: "Uma seqüência $\{x_n\}_{0<n<\infty}$ é considerada uma seqüência de Cauchy em um espaço métrico $(\boldsymbol{X},d)$ se $\{x_n\} \in \boldsymbol{X}$ e, para qualquer $\varepsilon > 0$ e $\varepsilon \in \boldsymbol{\Re}$, 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 essa seqüência converge para um ponto, pois esse ponto pode não existir dentro do espaço $\boldsymbol{X}$. Um exemplo de uma seqüência de Cauchy que não converge para um ponto é: dado o espaço $\boldsymbol{X} = (0;1) = \{x \mid 0<x<1, \ x \in \boldsymbol{\Re}\}$ e a seqüência de Cauchy $\{x_n=1/n\}_{1<n<\infty} \subset \boldsymbol{X}$, pode-se facilmente observar que:

$$\lim_{n \rightarrow \infty} \left(\frac{1}{n} \right) = 0 \notin (0;1). \tag{9.4}$$

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

$$x_f = \lim_{n\to\infty} x_n . \tag{9.5}$$

É relativamente fácil de se observar que toda seqüência convergente é uma seqüência de Cauchy (prova A_1 , Apêndice A) [55]. A partir disso, pode-se, então, definir o espaço métrico completo.

Espaços Métricos Completos

Definição 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 $x_f \in \boldsymbol{X}$ ".

Dessa forma, dada uma seqüência de Cauchy em um espaço métrico completo, então essa seqüência converge para um ponto limite x_f pertencente a esse espaço métrico de forma que a bola aberta $B(x_f,\varepsilon)$, $\forall\varepsilon \in \Re > 0$ contém infinitos pontos da seqüência.

O Espaço Métrico ($\mathcal{H}$(X),h): espaço métrico onde os fractais existem

A definição do espaço métrico ($\mathcal{H}(\boldsymbol{X})$, h) é dividida em duas partes sendo a primeira a definição do espaço $\mathcal{H}$ e a segunda, a definição da métrica associada h. Em um espaço métrico completo ($\boldsymbol{X}$,d), seus subconjuntos compactos são fechados e limitados. Assim:

Definição 4: "Seja ($\boldsymbol{X}$,d) um espaço métrico completo. Então o espaço dos fractais $\mathcal{H}(\boldsymbol{X})$ é o espaço dos subconjuntos compactos e não vazios do espaço ***X*** ".

Dessa forma, cada ponto $A \in \mathcal{H}(\boldsymbol{X})$ é um subconjunto compacto do espaço métrico ($\boldsymbol{X}$,d), e, então, pode-se definir a distância (métrica associada) entre dois conjuntos A e B, sendo A e $B \in \mathcal{H}(\boldsymbol{X})$ como sendo a aplicação h: $\mathcal{H}(\boldsymbol{X})\times\mathcal{H}(\boldsymbol{X}) \to \Re$ tal que:

$$h = \max\left\{ \max_{x\in A}\left\{ \min_{y\in B}\{d(x,y)\}\right\} , \max_{y\in B}\left\{ \min_{x\in A}\{d(x,y)\}\right\}\right\} \tag{9.6}$$

Essa aplicação h é chamada *métrica de Hausdorff* e é, em geral, a métrica utilizada no espaço dos fractais $\mathcal{H}(\boldsymbol{X})$. Pode-se provar [56] que a aplicação h é uma métrica, pois obedece às quatro propriedades dadas em (9.1). Deve-se notar que a métrica h dos subconjuntos A, B de ($\mathcal{H}(\boldsymbol{X})$, h) é dependente da métrica $d(x,y)$ de ($\boldsymbol{X}$,d).

O passo final para completar a definição do espaço dos fractais é dado pelo seguinte teorema [1] [56]:

Teorema D: "Seja ($\boldsymbol{X}$,d) um espaço métrico completo. Então ($\mathcal{H}(\boldsymbol{X})$,$h$) é um espaço métrico completo. Além disso, como o espaço ($\mathcal{H}(\boldsymbol{X})$,$h$) é completo, se a seqüência de subconjuntos $\{A_n \in \mathcal{H}(\boldsymbol{X})\}_{0<n<\infty}$ é uma seqüência de Cauchy, então, ela converge. Dessa forma, existe um A_f tal que:

$$A_f = \lim_{n\to\infty} A_n \in \mathcal{H}(\boldsymbol{X}) \tag{9.7}$$

Formalizando, a seqüência de Cauchy para o espaço dos fractais é definida em [1] [56] como:

Definição 5: "Uma seqüência $\{A_n : n=1, 2, ..., \infty\} \subset \mathcal{H}(\boldsymbol{X})$ é uma seqüência de Cauchy se, dado um $\varepsilon > 0$ e $\varepsilon \in \mathfrak{R}$, existe um número inteiro positivo N tal que:

$$h(A_n,A_m) < \varepsilon, \quad \forall m,n>N \tag{9.8}$$

Assim, como o espaço de fractais ($\mathcal{H}(\boldsymbol{X})$, h) é completo, toda seqüência de Cauchy em $\mathcal{H}(\boldsymbol{X})$ converge para um ponto em $\mathcal{H}(\boldsymbol{X})$. Como será visto posteriormente, este resultado é muito importante para os algoritmos de geração de fractais.

9.2 - Definições Matemáticas: Mapas Contrativos e Construção de Fractais

Nesta seção são dadas as definições de transformação afim e dos dois principais teoremas da codificação fractal.

Transformações em espaços métricos e Transformações Afins

Primeiramente, uma transformação de um espaço métrico $\boldsymbol{X}$ em si mesmo é definida matematicamente por [55] [56]:

Definição 6: "Seja ($\boldsymbol{X}$,d) um espaço métrico. Uma transformação em $\boldsymbol{X}$ é uma função f: $\boldsymbol{X} \to \boldsymbol{X}$ que associa exatamente um ponto $f(x) \in \boldsymbol{X}$ a cada ponto $x \in \boldsymbol{X}$. Se f estabelece uma relação um-a-um entre um ponto $f(x)$ e x de forma que $f(x)=f(y) \Leftrightarrow x=y$, então é possível definir uma função $f^{-1} : \boldsymbol{X} \to \boldsymbol{X}$ 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 as transformações são funções definidas do espaço métrico $\boldsymbol{X}$ sobre si mesmo, então pode-se definir as aplicações sucessivas (iterações) da função f como sendo:

Definição 7: "Seja f: $\boldsymbol{X} \to \boldsymbol{X}$ uma transformação em um espaço métrico. As *iterações diretas* de f são as transformações $f^{on} : \boldsymbol{X} \to \boldsymbol{X}$ definido por $f^{o0}(x)=x$, $f^{o1}(x)=f(x)$,..., $f^{on}(x)= f(f^{o(n-1)}(x))$ para $n=0, 1, 2, \ldots$.

Se f for inversível, então as *iterações reversas* de f são as transformações:

$f^{o(-m)}: \boldsymbol{X} \rightarrow \boldsymbol{X}$ por $f^{o(-1)}(x)=f^{-1}(x),...,f^{o(-m)}(x)=(f^{om})^{-1}(x)$ para $m= 1, 2, 3, \ldots$.

Um exemplo de uma classe de transformações no espaço $(\mathfrak{R}, d_{L_2})$ são as transformações polinomiais [57][58]:

Definição 8: "Uma Transformação Polinomial de Grau N f: $\mathfrak{R} \rightarrow \mathfrak{R}$ é expressa na forma:

$$f(x)= a_0 + a_1.x^1 + a_2.x^2 + \ldots + a_N.x^N \tag{9.9}$$

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

Dentro das transformações polinomiais existe uma família de transformações que são chamadas ***TRANSFORMAÇÕES AFINS***, que são transformações polinomiais de grau 1. No caso acima, na forma: $f(x)=a.x+b$, onde $a, b \in \mathfrak{R}$.

Definição de Contratividade [7]

Definição 9: "Uma transformação $f : \boldsymbol{X} \rightarrow \boldsymbol{X}$ em um espaço métrico $(\boldsymbol{X},d)$ é chamada *contrativa* ou *mapeamento contrativo* se há uma constante real $0 \leq s < 1$ tal que:

$$d(f(x),f(y)) \leq s.d(x,y), \ \forall x,y \in \boldsymbol{X}, \tag{9.10}$$

onde o número s é chamado fator de contratividade de f". Nota-se que a aplicação iterativa do mapeamento contrativo gera uma seqüência de Cauchy não necessariamente convergente, pois o ponto ainda pode estar fora do espaço, já que a definição não diz se tratar de um espaço métrico completo. Com essa definição de contratividade, pode-se enunciar o seguinte teorema:

TEOREMA DO MAPEAMENTO CONTRATIVO

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

$$\lim_{n \rightarrow \infty} f^{on}(x) = x_f, \quad \forall x \in \boldsymbol{X} \text{".} \tag{9.11}$$

Pode-se notar que no caso, pela contratividade, a distância d em (9.10) vai diminuindo. Como se trata de um espaço métrico completo, essa distância convergirá para zero. Logo, os pontos x e y após as transformações iterativas convergirão para um determinado ponto no espaço. Esse ponto x_f é exatamente o já tão citado atrator.

TEOREMA DA COLAGEM

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

$$d(x, x_f) \leq \frac{1}{1-s} \cdot d(x, f(x)), \ \forall x \in \boldsymbol{X} \text{"}. \tag{9.12}$$

Pode-se notar que, no caso, a distância entre um ponto qualquer do espaço e seu atrator é menor do que a distância entre ele e sua transformação. Portanto, quanto menor $d(x,f(x))$, mais perto se está do atrator. Dessa forma, descobrindo-se a função $f(x)$ que leve $d(x,f(x))$ para zero, estar-se-á descobrindo a função associada a esse atrator x_f (prova A_2 , Apêndice A).

MAPEAMENTOS CONTRATIVOS NO ESPAÇO DOS FRACTAIS

Até este ponto foram definidos mapeamentos contrativos em espaços métricos genéricos $\boldsymbol{X}$. 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 esse problema em [56] apresenta-se o seguinte lema que analisa a relação entre um mapeamento contrativo $\omega : \boldsymbol{X} \to \boldsymbol{X}$ e o seu equivalente no espaço ($\mathcal{H}$ ($\boldsymbol{X}$), h).

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

$$\omega(A) = \{\omega(x) \mid x \in A\}, \quad \forall A \in \mathcal{H}(\boldsymbol{X}) \tag{9.13}$$

é um mapeamento contrativo em ($\mathcal{H}$ ($\boldsymbol{X}$), h) com fator de contratividade s " (prova A_3 , Apêndice A).

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

Dadas essas ferramentas matemáticas, pode-se, agora, definir o processo de construção de fractais denominado sistemas de funções iterativas – IFS que é o princípio básico do processo de compressão fractal de imagens.

9.3 – Sistema de Funções Iterativas (IFS – *Iterated Function Systems*)

Nos anos 80, Michael Barnsley apresentou idéias sobre teorias fractais. Outros pesquisadores já haviam mostrado que sistemas caóticos eram capazes de produzir imagens fascinantes, conhecidas como "atratores estranhos". Barnsley , entretanto, foi o primeiro a resolver o problema inverso: dada uma imagem específica, seria possível determinar o sistema de mapeamentos que tem essa dada imagem como atrator?" Barnsley usou um sistema particular de mapeamentos que chamou de Sistema de Funções Iterativas (IFS – *Iterated Function Systems*).

Uma imagem tal como o fractal "folha de samambaia" da Fig. 8.1 do Capítulo 8 pode ser reproduzido com um IFS relativamente simples devido à sua auto-similaridade global. Isto é, a imagem inteira é feita de pequenas cópias de si mesma.

IFS é, na melhor hipótese, a forma mais crua de compressão fractal de imagens, diferentemente do PIFS (Sistema de Funções Iterativas Particionadas – *Partitioned Iterated Function Systems*) que melhor se adequa às imagens reais, sendo efetivamente utilizado nos codificadores fractais atuais [1]. Contudo, a compreensão do IFS é essencial para se compreender o modo como a compressão fractal trabalha [59].

Um sistema de funções iterativas (IFS) em R^2 consiste de uma coleção de transformações contrativas $\{\omega_i : R^2 \rightarrow R^2 \mid i=1,...n\}$ que mapeiam o plano R^2 para si mesmo em cada iteração. Essa coleção de transformações define o mapa [3]:

$$W(\cdot) = \bigcup_{i=1}^{n} \omega_i(\cdot) \tag{9.14}$$

Tratando a imagem inicial como um subconjunto A do plano (x,y), pode-se calcular ω_i (A) para cada i, e fazendo a união dos resultados, obtém-se um novo conjunto W(A), representando a imagem resultante ao fim de uma iteração. Pode-se ilustrar isso através da Fig. 8.4 onde, por exemplo no primeiro caso, a imagem resultante ao fim de uma iteração é composta pelas 3 cópias reduzidas da imagem original.

Hutchinson trabalhou teoremas acerca da contratividade de W [53] e colocou que se o mapa W (conjunto de transformações afins) for contrativo, terá um único ponto fixo no espaço de todas as imagens, isto é, pode-se aplicar repetidamente W à qualquer imagem (ou conjunto) inicial, que se converge para uma imagem fixa. Logo, W (ou os ω_i) determinam completamente uma única imagem [3].

Assim, dada uma imagem inicial A_0, aplicando uma vez o conjunto de transformações W, gera-

se A_1=W(A_0); a segunda vez gerando A_2=W(A_1)=W(W(A_0))≡W^{o2}(A_0) e assim por diante. Dessa forma, o atrator é o conjunto limite:

$$A_f = \lim_{n\to\infty} W^{on}(A_0) \tag{9.15}$$

que independe da escolha de A_0. Esse modelo é adequado para imagens bi-tonais (preto e branco) pois está-se trabalhando em $\{\omega_i : R^2 \to R^2 \mid i=1,...n\}$.

Para se trabalhar com imagens em escala de cinza é assumido um modelo matemático que trata a imagem como uma função contínua e limitada no espaço. Para uma imagem quadrada, é interessante limitar-se o seu domínio à um quadrado unitário. A última restrição é considerar também a amplitude como um valor contínuo limitado ao intervalo [0,1].

Assim, pode-se referir à imagem como a função $z = f(x,y)$ tal que [3]:

$$(x,y) \in \{(u,v) : 0 \le u,v \le 1\} \equiv I^2 \quad \text{e} \quad f(x,y) \in I \equiv [0,1] \tag{9.16}$$

sendo que I é o intervalo $[0,1]$ e I^2 é o quadrado unitário.

Formalizando matematicamente esses conceitos colocados, tem-se que na seção 9.1 e 9.2, havia sido apresentado o conceito de contratividade para um mapeamento simples no espaço de fractais $\mathcal{H}(\boldsymbol{X})$. No entanto, agora, essa definição pode ser entendida para os citados conjuntos de mapeamentos contrativos (W) de forma a possibilitar outras maneiras para se gerar curvas fractais. O lema a seguir, cuja prova é apresentada em [56], define a contratividade de conjuntos de mapeamentos contrativos.

CONTRATIVIDADE DE W(A):

Lema D: "Seja ($\boldsymbol{X}$, d) um espaço métrico completo e $\{\omega_n : \boldsymbol{X} \to \boldsymbol{X}$, $n=1, 2, ..., N\}$ um conjunto de mapeamentos contrativos neste espaço, e portanto, em ($\mathcal{H}(\boldsymbol{X})$, h). Seja o fator de contratividade de ω_n dado por s_n (Definição 9) para cada n. Definindo $W : \mathcal{H}(\boldsymbol{X}) \to \mathcal{H}(\boldsymbol{X})$ por:

$$W(A) = \omega_1(A) \bigcup \omega_2(A) \bigcup \ldots \bigcup \omega_N(A) \tag{9.17}$$

$$W(A) = \bigcup_{n=1}^{N} \omega_n(A), \text{ para cada } A \in \mathcal{H}(\mathbf{X}). \tag{9.18}$$

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

Com o lema define-se os sistemas de funções iterativas (IFS – *Iterated Function Systems*) [56]:

DEFINIÇÃO E PROPRIEDADES DE SISTEMAS DE FUNÇÕES ITERATIVAS (IFS):

Definição 10: "Um Sistema de Funções Iterativas – IFS – consiste em um espaço métrico completo ($\boldsymbol{X}$,d) e um conjunto finito de mapeamentos contrativos $\omega_n : \boldsymbol{X} \to \boldsymbol{X}$, com os respectivos fatores de contratividade s_n, para $n=1, 2, ..., N$. A notação para a IFS é $\{(\boldsymbol{X}, d); \omega_n, n=1, 2, ..., N\}$ e seu fator de contratividade é $s = max\ \{s_n , n=1, 2, ..., N\}$".

O teorema a seguir resume as principais propriedades dos sistemas de funções iterativas e define o conceito de *atrator* de um IFS [56].

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

$$W(A) = \bigcup_{n=1}^{N} \omega_n(A) \tag{9.19}$$

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

$$h(W(A), W(B)) \leq s \cdot h(A, B), \tag{9.20}$$

para todo $A,B \in \mathcal{H}(\boldsymbol{X})$. Seu único ponto fixo, $A_f \in \mathcal{H}(\boldsymbol{X})$ e chamado de *atrator* da IFS, é dado por (teorema do ponto fixo de mapas contrativos):

$$A_f = \lim_{n \to \infty} W^{on}(A), \quad \forall A \in \mathcal{H}(\boldsymbol{X}), \tag{9.21}$$

de forma que $A_f = W(A_f)$. A distância entre um dado $A \in \mathcal{H}(\boldsymbol{X})$ e o atrator A_f obedece à inequação (TEOREMA DA COLAGEM):

$$h(A, A_f) \leq \frac{1}{1-s} h(A, W(A))" \tag{9.22}$$

Esse 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).

9.4 – Comentários

Neste capítulo, foram apresentados os princípios matemáticos da compressão fractal, com destaque para os teoremas da colagem e do mapeamento contrativo. O primeiro deles permite que, dado

o atrator, possa-se determinar os coeficientes das transformações afins a serem enviados (codificação fractal). O segundo garante que o processo fractal converge para um atrator (decodificação fractal).

Colocada a base matemática, foi também apresentado o método de geração de fractais conhecido como IFS (*Iterated Funcion Systems*), que é a inspiração do sistema de codificação fractal atual de imagens denominado PIFS (*Partitioned Iterated Funcion Systems*). O PIFS será abordado detalhadamente no Capítulo 10, dada a sua importância e por ser o sistema utilizado na parte fractal do codificador híbrido proposto neste trabalho.

Capítulo 10

Sistemas de Funções Iterativas Particionadas - PIFS

Uma imagem real, em geral, não apresenta a auto-similaridade semelhante à da folha de samambaia da Fig. 8.1, dita "auto-similaridade global". Ao invés disso, algumas de suas áreas apresentam similaridades entre si (similar por partes). Para codificar essas imagens reais é necessária a aplicação do PIFS (Sistema de Funções Iterativas Particionadas – *Partitioned Iterated Function Systems*), método efetivamente usado nos codificadores fractal atuais [1] e abordado neste capítulo.

10.1 – Introdução ao PIFS

Uma imagem real como a mostrada na Fig. 10.1 não apresenta o tipo de auto-similaridade que pode ser encontrado nos fractais da Fig. 8.1, ou seja, a imagem não parece conter transformações afins de si mesma. No entanto, a imagem possui algumas áreas que apresentam similaridades. Na Fig.10.1b podem ser vistas similaridades entre as regiões destacadas no chapéu e espelho e entre regiões destacadas no ombro [3].

(a) (b)

Fig. 10.1 – (a) Imagem Lena Original (256 x 256 pixels);
(b) Exemplos de similaridades nessa imagem. [3]

A distinção entre esse tipo de similaridade e o dos fractais apresentados na Fig. 8.1 é que, naquele caso, a imagem era formada por cópias transformadas de si mesma *inteira* e aqui a imagem é formada por cópias propriamente transformadas de *partes* de si mesma. Dessa forma, as partes transformadas geralmente não se encaixam perfeitamente. Por conseqüência, a imagem codificada pelo PIFS não será uma cópia idêntica da imagem original, mas sim uma aproximação dela.

10.2 – Decodificação e Codificação do PIFS

10.2.a - Decodificação PIFS:

Usualmente, para o PIFS, se define inicialmente o processo de decodificação. Assim sendo, uma vez transmitidos os dados, cada iteração do processo de *decodificação* de imagens naturais em escala de cinza através do PIFS é realizado da seguinte forma: uma máscara seleciona uma parte de uma imagem no decodificador (essa parte é referenciada como D_i – *"domain-block"*ou bloco-domínio), na qual aplica-se uma subamostragem e uma transformação-afim ω_i *(*já transmitida). A seguir, esse *domain-block* já subamostrado e transformado é copiado para uma região específica desta mesma imagem (região esta que é referenciada por R_i – *"range-block"* ou Bloco-Imagem). Esse processo é repetido para todas as partes da imagem. O conjunto de todos os *domain blocks* é chamado de *Domain pool*. O conjunto final de todos os *range-blocks* formará a imagem já decodificada após esta iteração.

Assim, para realizar a codificação e decodificação através do PIFS é necessária primeiramente a definição de quatro aspectos básicos [56]:

- O número de cópias transformadas que irão compor a imagem de saída, ou seja, o número de *range-blocks*. Em geral essa informação é transmitida de forma implícita, como no caso da partição *quadtree* que será apresentada no Capítulo 11;
- A máscara que selecionará qual a parte afetada (*domain-block*) em cada transformação.
- O ajuste de brilho e contraste para cada *domain-block*;
- Os demais coeficientes das transformações afins (além de contraste e brilho) associados a cada *domain-block*.

Dada uma imagem f em escala de cinza segundo (9.16), uma iteração do processo de decodificação utilizando uma partição com N partes (N *range-blocks*) pode ser descrito como a aplicação do mapeamento W da seguinte forma [56]:

$$W(f) = \omega_1(f) \cup \omega_2(f) \cup ..... \cup \omega_N(f) \tag{10.1}$$

onde ω_i é aplicada *somente* sobre a respectiva região D_i. É importante salientar que associado a cada ω_i tem-se uma região D_i na imagem de entrada e uma região R_i na imagem de saída.

Como $W(f)$ representa uma imagem, deve-se salientar que $\cup R_i = I^2$ e que $R_i \cap R_j = \varnothing$, se i≠j, ou seja, R_i é uma *partição* da imagem de saída (I^2). Dessa forma, pode-se dizer que o processo iterativo é tal que a saída da primeira iteração f_1 é dada por $f_1=W(f_0)$, a da segunda $f_2=W(f_1)=W(W(f_0))=W^{o2}(f_0)$ e

assim por diante.

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 [3], essa prova é impossível no sentido completamente geral.

No entanto, para imagens reais se o mapa *W* for contrativo, segundo o teorema de Hutchinson, há um ponto fixo tal que *W(f)=f*; ou seja, existe o atrator:

$$f_\infty = \lim_{n\to\infty} W^{on}(f_0) \tag{10.2}$$

Definição 11: "Seja (***X***,*d*) um espaço métrico completo, e D_i, $R_i \subset \boldsymbol{X}$, para $i = 1, 2, ..., n$ tal que:

$$\bigcup_{i=1}^{n} R_i = I^2 . \tag{10.3}$$

Um sistema particionado de funções iterativas é uma coleção de mapas contrativos, $W=\{\omega_i : D_i \to R_i \mid i=1, 2, ..., n\}$".

10.2.b - Codificação PIFS:

Codificar uma dada imagem *f* significa *encontrar a coleção de mapas* $\omega_1, \omega_2, ..., \omega_N$ com $W = \bigcup_{i=1}^{N} \omega_i$ tal que $f = f_\infty$, ou seja, tal que a imagem *f* que deseja-se codificar seja o ponto fixo do mapeamento *W*.

De forma geral, a codificação fractal consiste em encontrar as auto-similaridades dos blocos da imagem dentro de si mesma em escala maior. Assim, a cada bloco da imagem a ser codificada, (*domain-block*), são aplicadas subamostragem, filtragem de média e transformações-afins ω_i. Os resultados de todas as transformações são comparados com cada bloco desta mesma imagem e de tamanho menor que o tamanho dos *domain-blocks*. Estes blocos de tamanho menor são chamado de *range-blocks*. Uma vez encontrado o melhor *matching* segundo um critério pré-determinado, ou seja, uma vez encontrada a auto-similaridade, somente serão enviados basicamente ao decodificador os coeficientes da transformação afim responsável pela formação de cada *range-block* e o endereço do *domain-block* que foi escolhido como seu equivalente.

A necessidade de se encontrar o melhor *matching* surge uma vez que, em geral, a intenção de determinar a região D_i da imagem que após a transformação gere exatamente R_i não pode ser alcançada. Isso ocorre porque uma imagem real dificilmente é composta somente por partes que, após a transformação, se encaixem perfeitamente em outra parte da mesma imagem.

Assim, o que sempre é possível é tentar encontrar uma outra imagem *f'* tal que $f' = f_{\infty}$ e que a métrica *d(f, f')* seja pequena o suficiente, ou seja, procura-se o mapa *W* tal que o seu ponto fixo *f'* seja próximo (ou se pareça com) a imagem *f*. A métrica *d(f, f')* pode ser definida de várias formas, mas uma forma simples e bastante efetiva é a função *sup*

$$d(f(x,y), g(x,y)) = máx|f(x,y) - g(x,y)|,\ (x,y) \in I^2 \tag{10.4}$$

O processo de codificação, então, se resume a encontrar as regiões D_i e os mapas ω_i tais que ao aplicar ω_i em D_i obtém-se algo mais próximo possível a região R_i. Num caso simples, a determinação de R_i e D_i é arbitrária como no exemplo da Fig. 10.2 em que os R_i foram escolhidos como sendo blocos de 8x8 pixels distintos da imagem e D_i blocos de 16x16 não necessariamente distintos (possível *overlap*).

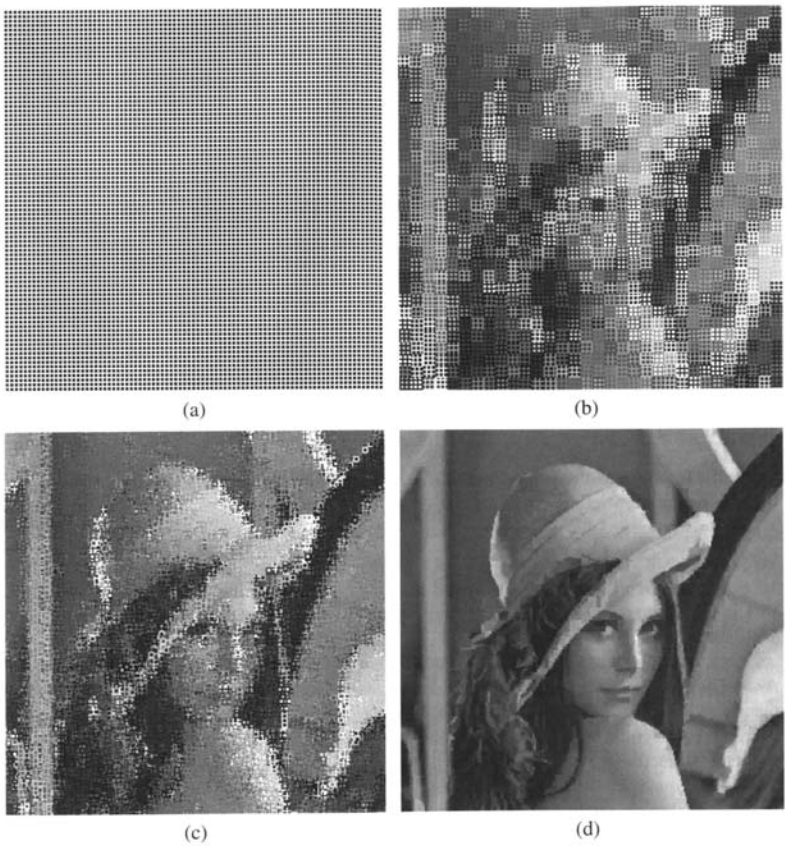


Fig.10.2 – Decodificação fractal da imagem Lena: a) imagem inicial arbitrária; b) após a 1ª iteração; c) após a 2ª iteração; d) após a 10ª iteração. [56]

O PIFS pode ser visto como uma generalização do IFS. Essa generalização simplifica (e torna possível na maioria dos casos) a codificação de conjuntos não auto-similares, como as imagens naturais.

No IFS apresentado no Capítulo 9 viu-se que a decodificação independe da imagem inicial. No PIFS isso nem sempre é verdade. Sendo os domínios restritos, 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 (caso relevante para este trabalho), esse 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.

10.2.c - Formalização Matemática do PIFS:

O primeiro passo na codificação de imagens com PIFS é a definição do modelo matemático que será usado para a imagem natural:

Definição 12: "Seja $I = [0;1] \subset \Re$ 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 $\boldsymbol{F} = \{f \mid f: I^2 \rightarrow \Re\}$".

Deve-se notar que a função f mapeia o quadrado unitário para o intervalo unitário, mas considera-se que a imagem de f seja $\Re$ para que as somas e subtrações de imagens fiquem bem definidas. Adotando a métrica do supremo d_{sup} para este espaço, Rudin [60] provou que o espaço métrico ($\boldsymbol{F}$, d_{sup}) é um espaço métrico completo. Embora o *range* e o *domain* da função sejam, respectivamente, R_i x I e D_i x I, é usual referenciá-los apenas como *range-block* R_i e *domain-block* D_i.

Generalizando a equação (8.2) para o caso de imagens com tons de cinza e definido o modelo de imagem, suponha que se deseja codificar por PIFS uma imagem f. Ou seja, deseja-se encontrar a coleção de mapas $\omega_1, \omega_2, ..., \omega_n$ com:

$$\omega_i \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ o_i \end{bmatrix}, \; i=1, 2, ..., n \tag{10.5}$$

onde $\{a_i, b_i, c_i, d_i, e_i, f_i\}$ define a transformação geométrica; s_i determina o contraste e o_i o brilho. Todos esses componentes compõem a transformação afim, mas por vezes, a transformação geométrica é referenciada separadamente, uma vez que é transmitida separadamente do brilho e contraste no *bitstream* do sinal codificado. A transformação geométrica é enviada em 3 bits. Tipicamente o brilho é enviado em 7 bits, e o contraste em 5 bits. Maiores detalhes podem ser encontrados em [3].

$W(f)$ é definido por:

$$W(f) = \bigcup_{i=1}^{N} \omega_i (f \cap (D_i \times I)) \tag{10.6}$$

de modo que $f = f_\infty$ 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:

$$f_\infty = W(f_\infty) = \omega_1(f_\infty) \cup \omega_2(f_\infty) \cup \ldots \omega_n(f_\infty). \tag{10.7}$$

O objetivo, na prática, é encontrar $f' = f_\infty$ tal que o erro $d_{sup}\ (f, f')$ seja pequeno o suficiente, conforme já colocado. Nesse caso:

$$f \approx f' = W(f') \approx W(f) = \omega_1(f) \cup \omega_2(f) \cup \ldots \omega_n(f) \tag{10.8}$$

Então, é suficiente aproximar os blocos-R_i dos blocos-D_i transformados minimizando a expressão:

$$\min_{\omega_i} \{d_{sup}(f \cap (R_i \times I), \omega_i(f))\}, \quad i = 1, \ldots, n \tag{10.9}$$

10.2.d - Transformações Afins:

Para facilitar a codificação, os mapas ω_i são, em geral, selecionados a partir de um conjunto limitado de possíveis mapas. Na prática da codificação, divide-se cada mapa (10.5) em duas partes:

- Parte geométrica composta pelos coeficientes: a_i, b_i, c_i, d_i, e_i, f_i que escalonam, rotacionam, invertem e transladam o *domain-block* sobre o *range-block*;
- Ajuste de brilho/contraste: formado pelos coeficientes s_i e o_i.

Quando se utiliza partição *quadtree* (abordada na próxima seção), o formato de D_i e R_i é sempre quadrado e o tamanho do *domain-block* é, em geral, o dobro do tamanho do *range-block*. Assim, a escolha da parte geométrica do mapa é limitada a apenas uma redução de tamanho por um fator de 2 seguida de uma dentre as 8 opções: 4 rotações (0º, 90º, 180º e 270º) e 4 inversões (vertical, horizontal, diagonal principal, diagonal secundária). Assim, a minimização da equação (10.9) 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 *range-blocks* R_i e como se encontram os *domain-blocks* D_i correspondentes.

10.2.e - Particionamento da imagem: range-blocks e domain-blocks

O modo mais simples de particionar uma imagem é dividí-la em blocos de mesmo tamanho, à semelhança do que é feito nos esquemas tradicionais de compressão de imagem. No entanto, 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 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 seja mais eficiente.

Uma generalização da partição de tamanho fixo é a *partição quadtree*. Nesse esquema de particionamento, se para uma dada região R_i não for possível encontrar um D_i tal que o erro (10.9) seja menor que um erro máximo estipulado, então se subdivide essa região em quatro partes iguais processando cada uma separadamente.

Esse algoritmo é recursivo, ou seja, se qualquer uma dessas subdivisões também não alcançar o erro desejado, é subdividida em 4 partes e recomeça-se o processo. No caso *top-down* o processo de particionamento se inicia com a imagem inteira e segue recursivamente até que todas as partições atinjam o erro desejado.

O conjunto $\boldsymbol{D}$ de todos os possíveis *domain-blocks* (*Domain Pool)* pode ser escolhido como sendo todos os possíveis quadrados dentro da imagem, no entanto, é comum escolher-se $\boldsymbol{D}$ como um subconjunto desse espaço [3]. Um exemplo de $\boldsymbol{D}$ é o conjunto de quadrados de tamanho fixo dentro de uma grade com espaçamento de metade do tamanho de cada D_i [61]. É conveniente a escolha do tamanho de D_i como sendo o dobro do tamanho do cada R_i.

Na determinação de $\boldsymbol{D}$ da partição *quadtree*, não é levado em conta o contexto da imagem, o que faz com que a coleção de possíveis *domain-blocks* precise ser grande para tornar possível o encontro de uma boa aproximação para um dado *range-block* R_i. Apesar dessa característica, esse tipo de partição apresenta ótimos resultados, sendo bastante simples e eficiente para a codificação fractal.

10.2.f - Tipos de codificadores PIFS

Como os esquemas de partição são em geral adaptativos, o número de *range-blocks* R_i não é fixo. Quanto maior o número de *range-blocks*, maior a fidelidade, mais transformações e, conseqüentemente, menor a compressão. Esse balanceamento entre fidelidade e taxa de compressão leva a duas possíveis aproximações para a codificação: visando uma determinada compressão ou uma determinada fidelidade. O fluxograma de um esquema básico buscando uma fidelidade mínima e_c é mostrado na Fig. 10.3 [3].

Sendo $r_{mín}$ o parâmetro que designa o tamanho mínimo permitido ao *range-block* R_i, pode-se notar que apesar de o parâmetro e_c ser a fidelidade mínima para cada *range-block*, esse valor não garante que a imagem codificada tenha essa fidelidade, uma vez que a fidelidade depende também do valor de

$r_{mín}$ e do conjunto domínio. No entanto, quanto menor o e_c, melhor a fidelidade da imagem codificada.

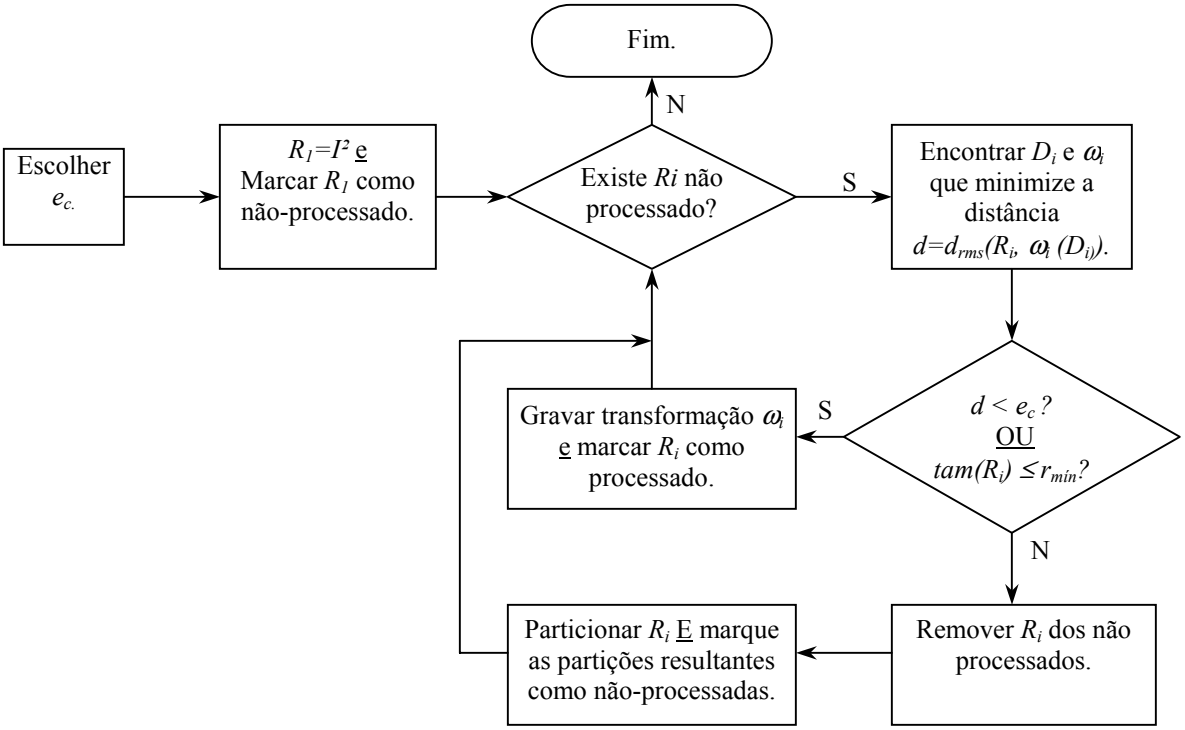


Fig. 10.3 – Fluxograma do codificador fractal de imagens com fidelidade almejada [3]

10.3 – Comentários

Neste capítulo foi apresentado o sistema de codificação fractal conhecido como PIFS (Sistema de Funções Iterativas Particionadas – *Partitioned Iterated Function Systems*), método de codificação efetivamente usado nos codificadores fractal atuais [1], incluindo no Codificador Híbrido desenvolvido neste trabalho. Foram apresentados seus princípios de funcionamento (baseado na similaridade por partes), o particionamento da imagem em *domain-blocks* e *range-blocks*, a aplicação das transformações afins, o método básico de busca e encontro do *domain-block*-equivalente e as informações a serem enviadas ao decodificador.

O capítulo seguinte apresentará os dois codificadores fractais que serão utilizados como sistemas-base para comparação com o codificador híbrido proposto nesta pesquisa. Trata-se do Codificador Fractal com Acelerador, de Fisher [3]; e do Codificador Fractal de Cardinal [4].

Capítulo 11

Codificação Fractal Acelerada

A codificação fractal consiste em representar os blocos da imagem através dos coeficientes de transformações contrativas, explorando-se o conceito de auto-similaridade. Esse tipo de compressão apresenta ótima qualidade de imagens, mesmo a baixas taxas de bits. Entretanto, cada *range-block NxN* é comparado a *todos* os *domain-blocks 2Nx2N* da *domain pool* dentro de uma imagem *MxM*. Como conseqüência, o algoritmo de *matching* possui complexidade computacional $O[M^4]$, o que exige um tempo exaustivo de processamento, tipicamente levando horas. Esse peso computacional é o principal motivo de hesitação da adoção de aplicações fractais.

Basicamente, as tentativas de acelerar a codificação fractal consistem em modificar os seguintes aspectos: a composição da *domain pool*, o tipo de procura usado no *block matching*, ou a representação/quantização dos parâmetros transformados. Trabalhos recentes podem ser encontrados em [4][62]-[71].

Neste capítulo são apresentados os dois codificadores fractais que serão utilizados como Sistemas-Base de comparação com o codificador Híbrido proposto nesta pesquisa. O primeiro deles, o Codificador Fractal com Acelerador (Fisher) [3], foi escolhido por ser um dos codificadores mais implementados na literatura científica atual. O segundo, o Codificador Fractal de Cardinal [4] foi também escolhido por ser uma referência bastante atualizada e requisitada como um bom parâmetro de comparação pela revista científica *IEEE Transactions on Image Processing*. Este capítulo também apresenta referências de outros trabalhos na área de aceleração da codificação fractal de imagens.

11.1 – Codificador Fractal de Fisher

Fisher [3] desenvolveu uma classificação de domínios com o objetivo de reduzir o número de comparações de *matching*. O ponto principal desse esquema é que, primeiramente, todos os *domain-blocks* são classificados. Durante a codificação, cada *range-block* é classificado e somente será comparado com os *domain-blocks* de mesmo tipo que o seu (segundo um critério baseado em média e variância), o que significativamente reduz o número de comparações de *matching*. Por esse motivo, esse esquema é referenciado como “acelerador de Fisher”.

A seguir serão apresentados os princípios desse codificador tais como a partição *quadtree*, até se chegar à citada classificação de blocos propriamente dita.

11.1.a - Determinação dos blocos-imagem

A etapa de codificação do esquema de compressão fractal de Fisher segue o fluxograma apresentado na Fig. 10.3. Note que a métrica utilizada é a métrica *rms*, escolha que facilitará a determinação dos valores de s_i e o_i, conforme será colocado à frente. Nesse esquema, a coleção de *range-blocks* é gerada a partir do sistema de partição *quadtree* e a coleção de *domain-blocks* é formada pelos blocos quadrados da imagem (também gerados por *quadtree*), cujo tamanho é duas vezes o tamanho do *range-block*.

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 desse quadrado). A raiz da árvore é a imagem original, como no exemplo da Fig. 11.1.

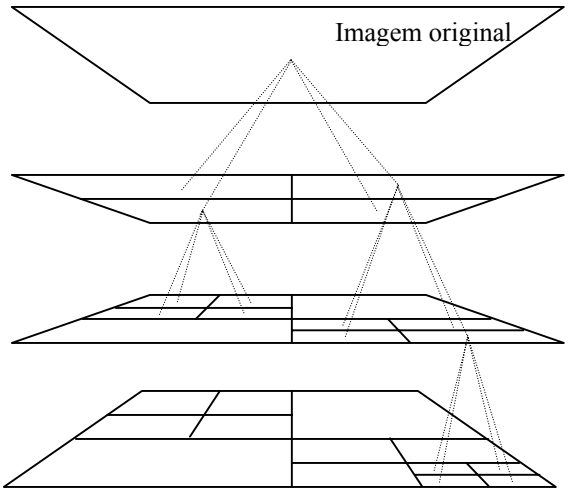


Fig. 11.1 – Exemplo de partição *quadtree*

Os *range-blocks* 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 (*range-block*) será então comparado com todos os *domain-blocks* pertencentes à *Domain Pool* que tenham o dobro do tamanho do *range-block*. Estes *domain-blocks* são os candidatos a *domain-block*-equivalente.
- Para a comparação citada acima, cada candidato a *domain-block* é primeiramente subamostrado por um fator de 2, onde cada *pixel* é substituído pela média de dois *pixels* consecutivos. O objetivo é que os candidatos a *domain-block* tenham o mesmo tamanho do *range-block* para permitir a comparação.
- Aplicam-se as transformações afins otimizadas a cada candidato a *domain-block*-equivalente e calcula-se a d_{rms} deste com o *range-block* correspondente.

- 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 da *quadtree* for menor que o máximo permitido (parâmetro do codificador: nível máximo da *quadtree*), particiona-se esse *range-block* em quatro partes e reinicia-se o processo. Caso contrário, o *domain-block* que gere a menor d_{rms} é escolhido como sendo o *domain-block*-equivalente . Segue a codificação do próximo *range-block*.

11.1.b - Determinação da Domain Pool D

O esquema de Fisher permite o uso de um dos três diferentes tipos de *Domain Pool* $\boldsymbol{D_1}$, $\boldsymbol{D_2}$ e $\boldsymbol{D_3}$. Nos três casos, os candidatos a *domain-blocks*-equivalentes são escolhidos como sendo blocos quadrados que tem o canto superior esquerdo alinhado com uma grade com passo *l*. Para cada tipo de *Domain Pool*, o espaçamento da grade é escolhido em função do tamanho requerido do *domain-block* como se segue:

- $\boldsymbol{D_1}$: *Domain pool* que contém um número aproximadamente igual de candidatos a *domain-blocks*-equivalentes em cada tamanho de bloco da *quadtree*. Espaçamento da grade igual a *l* em todos os tamanhos;

- $\boldsymbol{D_2}$: mais *domain-blocks*-candidatos pequenos e menos *domain-blocks*-candidatos grandes, privilegiando a codificação dos detalhes das imagens. Espaçamento da grade igual ao tamanho do bloco dividido por *l*. Esse foi o esquema proposto por Jacquin [61];

- $\boldsymbol{D_3}$: mais *domain-blocks*-candidatos grandes e menos *domain-blocks*-candidatos pequenos, proporcionando assim, uma melhor codificação dos *range-blocks* grandes, pois esses oferecem maior compressão (menor número de transformações a serem transmitidas/armazenadas). O espaçamento da grade é inverso ao anterior, ou seja, o maior tamanho possível de bloco tem uma grade com espaçamento igual ao tamanho do menor bloco dividido por *l* e vice-versa.

É interessante notar que, para cada candidato a *domain-block*-equivalente, serão testadas as 8 possíveis orientações (transformações geométricas do Capítulo 10) o que faz com que, na verdade, cada candidato seja testado 8 vezes, sendo que cada teste tem um par (s_i, o_i) próprio calculado como mostrado no próximo item.

11.1.c - Cálculo dos valores s_i e o_i

Sejam $\boldsymbol{r}_i$ o *range-block* transformado em vetor; $\boldsymbol{d}_i$ o *domain-block* transformado em vetor; $\boldsymbol{1}$ o vetor unitário; e *NxN* as dimensões do *range-block*. Para calcular os valores de s_i e o_i ótimos para cada

candidato a *domain-block*-equivalente em uma dada transformação geométrica utiliza-se a expressão:

$$\min_{s_i\, o_i}\left(d_{rms}\left(\mathbf{r}_i,\ s_i\cdot\mathbf{d}_i+o_i\cdot\mathbf{1}\right)\right)=\left(\sum_{j=1}^{N^2}\left|r_{ij}-\left(s_i\cdot d_{ij}+o_i\right)\right|^2\right)^{1/2} \tag{11.1}$$

Para resolver essa equação utilizar-se-á uma característica da métrica *rms*, na qual pode-se calcular a distância entre dois pontos utilizando o produto interno padrão (simbolizado pelo operador <·,·>) segundo a expressão:

$$d_{rms}(\boldsymbol{x},\boldsymbol{y})=\sqrt{\langle \boldsymbol{x}-\boldsymbol{y},\boldsymbol{x}-\boldsymbol{y}\rangle} \tag{11.2}$$

Assim:

$$d_{rms}^2\left(\mathbf{r}_i,\ s_i\cdot\mathbf{d}_i+o_i\cdot\mathbf{1}\right)=\left\langle \mathbf{r}_i-\left(s_i\cdot\mathbf{d}_i+o_i\cdot\mathbf{1}\right),\mathbf{r}_i-\left(s_i\cdot\mathbf{d}_i+o_i\cdot\mathbf{1}\right)\right\rangle=$$

$$=\ s_i^2\langle \boldsymbol{d}_i,\boldsymbol{d}_i\rangle+2o_is_i\langle \boldsymbol{d}_i,\boldsymbol{1}\rangle+o_i^2\langle \boldsymbol{1},\boldsymbol{1}\rangle-2s_i\langle \boldsymbol{d}_i,\boldsymbol{r}_i\rangle-2o_i\langle \boldsymbol{1},\boldsymbol{r}_i\rangle+\langle \boldsymbol{r}_i,\boldsymbol{r}_i\rangle \tag{11.3}$$

Diferenciando com relação a s_i e a o_i e igualando a zero obtém-se os valores:

$$s_i=\frac{\langle \boldsymbol{d}_i,\boldsymbol{1}\rangle\langle \boldsymbol{1},\boldsymbol{r}_i\rangle-\langle \boldsymbol{1},\boldsymbol{1}\rangle\langle \boldsymbol{d}_i,\boldsymbol{r}_i\rangle}{\langle \boldsymbol{d}_i,\boldsymbol{1}\rangle^2-\langle \boldsymbol{1},\boldsymbol{1}\rangle\langle \boldsymbol{d}_i,\boldsymbol{d}_i\rangle}\quad\text{e}\quad o_i=\frac{\langle \boldsymbol{d}_i,\boldsymbol{1}\rangle\langle \boldsymbol{d}_i,\boldsymbol{r}_i\rangle-\langle \boldsymbol{1},\boldsymbol{r}_i\rangle\langle \boldsymbol{d}_i,\boldsymbol{d}_i\rangle}{\langle \boldsymbol{d}_i,\boldsymbol{1}\rangle^2-\langle \boldsymbol{1},\boldsymbol{1}\rangle\langle \boldsymbol{d}_i,\boldsymbol{d}_i\rangle} \tag{11.4}$$

Substituindo pela definição do produto interno:

$$s_i=\frac{\sum \boldsymbol{d}_i\cdot\sum \boldsymbol{r}_i-N^2\cdot\sum \boldsymbol{d}_i\boldsymbol{r}_i}{\left(\sum \boldsymbol{d}_i\right)^2-N^2\cdot\sum \boldsymbol{d}_i^2}\quad\text{e}\quad o_i=\frac{\sum \boldsymbol{d}_i\cdot\sum \boldsymbol{d}_i\boldsymbol{r}_i-\sum \boldsymbol{r}_i\cdot\sum \boldsymbol{d}_i^2}{\left(\sum \boldsymbol{d}_i\right)^2-N^2\cdot\sum \boldsymbol{d}_i^2} \tag{11.5}$$

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 em termos de amplitude e precisarão ser quantizados. Para calcular o erro de aproximação usam-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). Caso o valor calculado ultrapasse s_{max}, o valor s_{max} é usado para o cálculo do erro.

11.1.d - Classificação dos blocos

(efetivamente o Acelerador de Fisher)

Este item apresenta o ponto principal do "acelerador de Fisher" [3]: a anteriormente citada classificação de blocos, concebida com o objetivo de reduzir o número de *domain-blocks* comparados com cada *range-block*. Para isso, nesse esquema primeiramente todos os *domain-blocks* são classificados. Durante a codificação, cada *range-block* também é classificado e somente será comparado com os *domain-blocks* de mesmo tipo que o seu. A idéia de desenvolver essa classificação foi desenvolvida independentemente em [61] e [72].

Muitas classificações são possíveis, mas Fisher utilizou a classificação por superclasse e subclasse apresentada em [72]: o sub-bloco quadrado é dividido em quatro quadrantes: esquerdo alto, direito alto, esquerdo baixo e direito baixo, numerados seqüencialmente. Para cada quadrante são calculados valores proporcionais à média e à variância dos *pixels*. Denotando os *n pixels* do quadrante *i* por r_1^i, ..., r_n^i, para *i = 1, 2, 3, 4*; a média e a variância são dadas em (11.6) em (11.7), respectivamente:

$$M_i = \frac{1}{n}\sum_{j=1}^{n} r_j^i \tag{11.6}$$

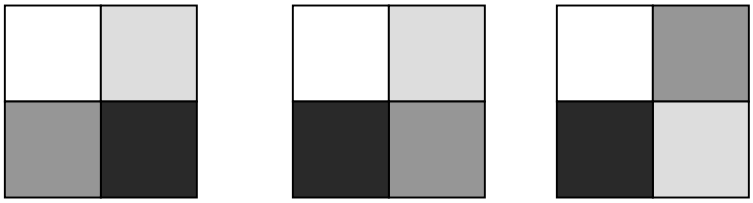
$$V_i = \frac{1}{n}\cdot\left(\sum_{j=1}^{n}\left(r_j^i\right)^2\right) - M_i^2 . \tag{11.7}$$

Pode-se, através de rotações e inversões, orientar o bloco de forma a ordenar as M_i, de modo que seja possível classificá-lo em uma das três superclasses de média (ilustradas na Fig. 11.2):

Superclasse 1) $M_1 \geq M_2 \geq M_3 \geq M_4$

Superclasse 2) $M_1 \geq M_2 \geq M_4 \geq M_3$

Superclasse 3) $M_1 \geq M_3 \geq M_2 \geq M_4$



Superclasse 1 Superclasse 2 Superclasse 3

Fig. 11.2 – Superclasses de classificação por brilho

Uma vez fixada a orientação, cada Superclasse de média tem 24 subclasses de variância consistindo nas 24 possíveis ordenações dos valores de V_i resultando 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. Em [3] são apresentados maiores detalhes sobre esta classificação, omitidos aqui por questão de objetividade.

11.1.e - Parâmetros de Codificação

Observando o acelerador de Fisher apresentado, tem-se os seguintes parâmetros de codificação:

- Os níveis mínimo e máximo da *quadtree*;
- O tipo de *Domain Pool* e o espaçamento da grade *l*;
- O limiar de tolerância *rms* ou fidelidade mínima: e_c;
- O máximo fator de brilho permitido s_{max};

A codificação de uma imagem consiste na transmissão dos seguintes dados:

- A partição *quadtree* final da imagem;
- Para cada *range-block*, a posição do *domain-block*-equivalente que é mapeado sobre ele;
- A orientação (transformação geométrica) usada para mapear os *pixels* do *domain-block*-equivalente sobre o *range-block*. Essa orientação vem claramente nesse esquema através da informação de superclasse e subclasse.
- Os valores s_i e o_i para cada *range-block*;

11.1.f - Armazenamento/transmissão

Para aumentar a compressão, Fisher utilizou a seguinte estrutura de *bitstream*:

- **Quadtree**: Um bit para cada nó da *quadtree* para indicar a subdivisão do nó atual (em caso de "quebra" nos 4 nós componentes, bit "1") ou o envio da informação de transformação (no caso de não haver "quebra", bit "0"). No último nível de codificação não é enviado nenhum bit, pois não pode haver mais subdivisões.
- **Escala e deslocamento**: Fisher usou cinco bits para armazenar o fator de escala s_i e sete bits para o deslocamento o_i. Resumidamente, será utilizada a notação $[s_i, o_i] = [5,7]$. Não foi utilizada nenhuma codificação adaptativa para estes coeficientes embora suas distribuições não fossem uniformemente distribuídas.
- ***Domain-Blocks***: Os *domain-blocks* são indexados e referenciados por seus índices. No entanto quando o fator de escala s_i é nulo, o *domain-block* é irrelevante e, portanto, não são armazenados

nem a posição do *domain-block*, nem a orientação.

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

11.1.g - Decodificação

Para imagens reais, que são as de interesse nesta pesquisa, o processo de decodificação consiste basicamente em realizar as iterações de W (Capítulo 10) a partir de *qualquer* imagem inicial. Cada iteração segue os seguintes passos:

1. A informação da partição *quadtree* é usada para gerar todos os *range-blocks*;
2. Para cada *range-block* R_i, o *domain-block*-equivalente D_i 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 posicionados no local do *range-block* correspondente com a orientação (transformação geométrica) indicada no código. As operações desse passo 3 compõem efetivamente a transformação afim.

Essas 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). Como se trata de um processo lento, o autor propõe ainda 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.

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.

Claramente, 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. Contudo, estes detalhes são auto-similares à imagem original, gerando aproximações interessantes como a mostrada na Fig. 11.3.

Como no processo de codificação a imagem é dividida em blocos disjuntos que são processados independentemente, a imagem reconstruída pode apresentar eventuais descontinuidades entre os *range-blocks*. Este efeito é conhecido como efeito de bloco. Para reduzi-lo, Fisher aplicou uma filtragem passa-baixas nos *pixels* das fronteiras dos blocos.

Como dito anteriormente, apesar deste codificador não ser otimizado, ele ainda é a referência de

comparação para muitos artigos da literatura científica recente em codificação fractal de imagens.

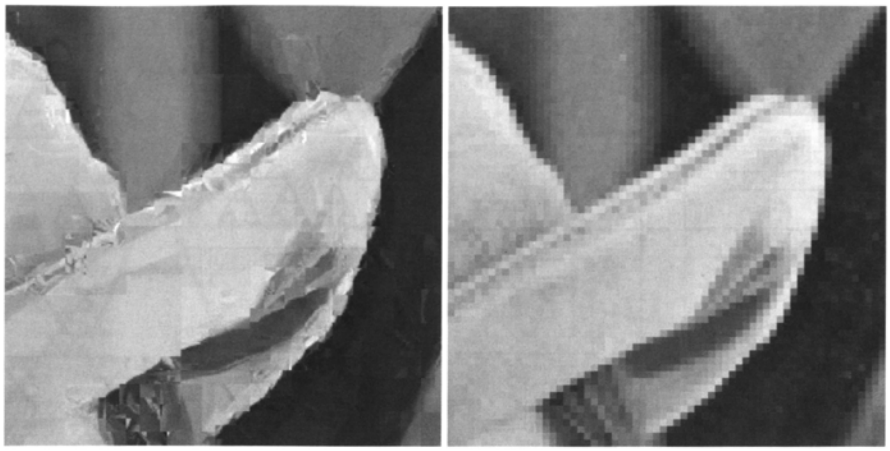


Fig. 11.3 – Ampliações da imagem original usando: (a) codificação fractal; (b) interpolação linear [3].

11.2 – Codificador Fractal de Cardinal

Cardinal [4] propôs um excelente algoritmo para compressão fractal de imagens em escala de cinza. Esse algoritmo faz uso de resultados anteriores de outros autores, reduzindo o processo de procura pelas transformações afins de cada *range-block* ao problema geométrico clássico de busca pelos vizinhos mais próximos num espaço euclidiano. Baseando-se na partição geométrica do espaço de características dos *range-blocks*, Cardinal conseguiu uma aceleração excelente na compressão sem perda de qualidade, quando comparado a diversos resultados anteriores.

O algoritmo de Cardinal usa como parâmetros os dois conjuntos de blocos já conhecidos: o conjunto dos *range-blocks* e o conjunto dos *domain-blocks*. Para cada um dos blocos é calculado um vetor de características (também chamado de "*key*"). O vetor adotado por Cardinal foi o operador projeção normalizada de Saupe, λ [3, Apêndice], dado por:

$$\lambda(x) = \frac{x - \langle x, e \rangle \cdot e}{\left\| x - \langle x, e \rangle \cdot e \right\|} \tag{11.8}$$

onde $e = (1,...,1)/\sqrt{k}$, sendo k o número de dimensões. Trata-se do espaço métrico (R^k, L_2).

Assume-se que todos os blocos possuam o mesmo tamanho, ou seja, os *domain-blocks* já sofreram a subamostragem com filtragem de média. Os blocos são tratados como simples vetores em R^k, vetores estes que são a saída de (11.8) . Deve-se notar que, aplicar o operador de Saupe, equivale a zerar a média do vetor e normalizá-lo.

Uma vez calculadas as *keys*, o próximo passo consiste em particionar recursivamente o espaço de procura, espaço esse que contém tanto as *range-keys* quanto as *domain-keys*. Este particionamento é

feito até que um número μ de *domain-keys* (pequeno o suficiente) seja deixado em cada subespaço. Cada *range-block* (correspondendo a uma *range-key* no subespaço) é então comparado a cada um dos *domain-blocks* (correspondendo às *domain-keys* remanescentes), e a melhor transformação para cada *range-block* é escolhida. Deve-se notar que a informação de particionamento não precisa ser enviada.

Às vezes, o processo pode terminar com um subespaço que não contenha nenhuma *domain-key*, somente *range-keys*, o que não faria sentido no processo de compressão. Assim, essa exceção precisa ser tratada separadamente, cancelando previamente a partição. Contudo, podem haver subespaços que contenham somente *domain-keys*. Nesse caso, os *domain-blocks* correspondentes a elas não serão utilizados para nenhuma comparação de *matching*.

O espaço de procura é recursivamente dividido em duas partes por um hiperplano *(k-1)*-dimensional. A cada passo, o hiperplano divisor é representado por um vetor normalizado $v \in R^k$ e por um valor escalar $\mu \in R$, onde o hiperplano é ortogonal a v e contém o ponto $\mu \cdot v$, como pode ser observado na Fig. 11.4 para o caso de R^2. A questão se concentra então em encontrar μ e v a cada passo para encontrar o hiperplano divisor correspondente.

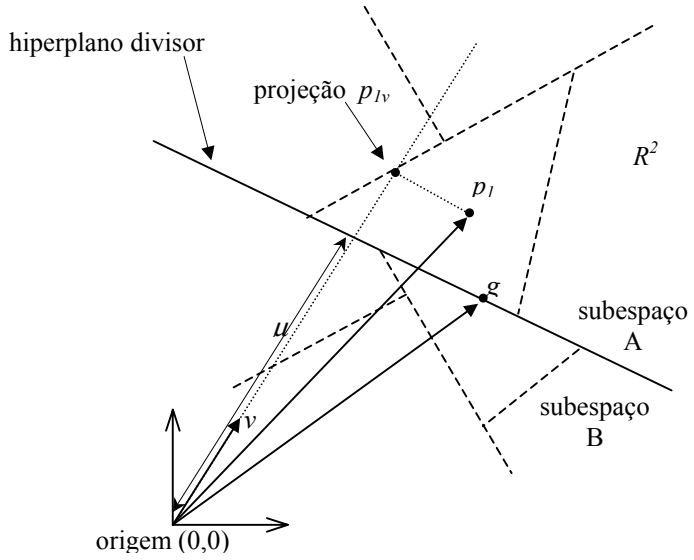


Fig. 11.4 – Ilustração do processo de particionamento: as linhas tracejadas representam os hiperplanos divisores subseqüentes. O par *domain-range* é testado somente se as *keys* correspondentes ficarem na mesma célula [4].

Chamemos as *keys* de p. Cada *key* p estará em um lado do hiperplano ou no outro, dependendo se a projeção $\langle p,v \rangle$ é maior ou menor que μ. Na Fig. 11.4, por exemplo, como $\langle p_1,v \rangle > \mu$ a *key* p_1 está no lado A do hiperplano.

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

nível, onde N_R e N_D são o número de *range-blocks* e *domain-blocks*, respectivamente.

Uma boa solução para encontrar v foi proposta em [73], onde é feita a matriz de covariância da distribuição das *keys*. A direção do plano é dada pelo principal autovetor (chamado v_{opt}) desta matriz. Essa solução foi usada como referência de comparação por Cardinal por considerá-la como sendo a solução ótima, uma vez que esta direção do plano maximiza a variância residual das *keys*, ou seja, maximiza a informação dos dois lados do divisor do hiperplano.

O processo até aqui de divisão de espaço de procura já era tradicional. Cardinal excelentemente propôs então um método rápido para encontrar μ e v. Assim, ele propôs uma heurística para encontrar um processo rápido de divisão do espaço sem ter que calcular o v_{opt} , uma vez que este processo é computacionalmente extensivo, especialmente em maiores dimensões. O método proposto por Cardinal para determinar o hiperplano divisor segue os seguintes passos:

- Calcula-se $\{r_i \in R^k \mid i = 1,..., N_R\}$ e $\{d_i \in R^k \mid i = 1,..., N_D\}$, que são o conjunto de *range-keys* e o conjunto de *domain-keys*, respectivamente.
- Calcula-se o centro de gravidade *g* da distribuição das *Range-keys*:

$$g = (1/N_R)\sum_{i=1}^{N_R} r_i \tag{11.9}$$

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

$$t_i = \frac{(r_i - g)}{\| r_i - g \|} \tag{11.10}$$

- Calcula-se

$$w = (1/N_R)\sum_{i=1}^{N_R} t_i \tag{11.11}$$

- 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 $\langle g, v \rangle$.

O objetivo dessa heurística é encontrar uma direção significativa na nuvem de *keys*. Primeiramente, Cardinal encontra o centro de gravidade das *range-keys* com o objetivo de centrar a distribuição, e então simplesmente tira a média das direções observadas a partir deste ponto até as *range-keys*. O fato de utilizar somente as *range-keys* para encontrar os hiperplanos divisores acelera muito o processo, uma vez que o número de *range-blocks* é muito menor do que o número de *domain-blocks*.

Uma vez que o hiperplano divisor tenha dividido o espaço de procura em células, é iniciado o

processo de busca pelo melhor *matching*, semelhante à Fisher [3] na Seção 11.1. Assim, dentro de cada célula, cada *range-block* será comparado com os *domain-blocks* componentes dessa mesma célula. Porém, Cardinal não usou as transformações geométricas fractais (Seção 10.2, Item "transformações afins"). Logo, para cada comparação *range-block* com *domain-block* é feito um único cálculo de S_i e O_i (brilho e contraste). A comparação que gere menor RMS fornecerá o *domain-block*-equivalente, semelhantemente a Fisher.

Os itens "parâmetros de codificação", "armazenamento e transmissão", e "decodificação" são basicamente iguais aos de Fisher, com as modificações pertinentes dadas a seguir.

No caso de "parâmetros de codificação", o tipo de *Domain Pool* é agora especificado pelas células geradas com os hiperplanos divisores e existe um novo parâmetro: o número máximo de *domain-keys* por célula. NA codificação não são usadas transformações geométricas. Em "armazenamento e transmissão" também não são enviados os bits de orientação.

Cardinal comparou sua proposta com 3 métodos [4]:

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 [4] validaram seu codificador como tendo desempenho superior aos demais métodos: proporcionou um aumento significativo de velocidade para a mesma qualidade de imagem. Os detalhes e parâmetros de codificação para a comparação com o codificador Híbrido Fractal-*Wavelet* proposto neste trabalho serão apresentados no Capítulo 12.

11.3 – Outros Codificadores Acelerados

Diferentes tentativas de reduzir o tempo de compressão fractal têm sido tentadas. As mais tradicionais dizem respeito à restrição da *Domain Pool* via pré-classificação dos blocos. Por exemplo, em [61] Jacquin classificou os blocos usando informações de borda e, em [3], Fisher pré-classificou os blocos segundo seus valores de média e variância (conforme apresentado na Seção 11.1).

Bogdan e Meadows [74] aplicaram uma rede auto-organizadora de *Kohonen* ao problema da codificação fractal, requerendo treinamento do codificador sobre a imagem a ser codificada. Esses autores reduziram o número de comparações *range-domain*, mas não a complexidade da comparação em si. McGregor *et al* [75], por outro lado, trabalhou esses dois problemas usando uma procura do tipo *K*-

D-tree nos *domain-blocks* combinada à extração de um pequeno número de características do bloco-imagem.

Um passo importante na questão da aceleração fractal foi dado em [65], e foi a inspiração do codificador de Cardinal apresentado na Seção 11.2. Em [65] Saupe reduziu a questão da procura pelo melhor *matching* à procura pelo vizinho mais próximo dentro de um espaço métrico conveniente. Nesta técnica, cada bloco é associado a um vetor de características tal que minimizar o erro de colagem entre o *range-block* e o *domain-block* equivale a minimizar a distância entre os vetores de característica correspondentes. A procura pelo melhor *matching* então consiste em simplesmente encontrar o vetor mais próximo no espaço vetorial. Deve-se colocar, entretanto, que o método de Saupe encontra o melhor *matching* e só depois calcula os valores de s_i e o_i, o que pode ferir a restrição de contratividade sobre s_i ($|s_i| < 1$). O método de Saupe pode ser feito de forma mais rápida se for usada uma estrutura *K-D-tree*.

Idéias similares são apresentadas em [76][77]. Também pode ser utilizada a quantização vetorial como em [73][78]. Em [52] um método ligeiramente menos sofisticado que o de Cardinal [4] é apresentado, mas segue a mesma filosofia. Cardinal argumenta que tais técnicas tendem a superar as aproximações clássicas que têm por base a redução da *Domain Pool*.

Em [62] é apresentada uma procura pelo vizinho mais próximo baseada na projeção ortogonal e pré-quantização dos parâmetros fractais transformados. Tong e Pi também desenvolveram um codificador de procura adaptativa que exclui um grande número de *domain-blocks* não qualificados [63] e Wang e Hsieh [64] elaboraram um método que explora a correlação entre *range-blocks* com o mesmo propósito. Bani Eqbal [69] propôs um método de procura em árvore para a *Domain Pool*.

Alguns algoritmos eficientes [68][70] foram também apresentados para aliviar a carga computacional mantendo a mesma qualidade de imagem do *Full Search*. Entretanto, pré-processamento é necessário para esses algoritmos.

Ao lado dessas tentativas de acelerar a codificação fractal pura, alguns codificadores híbridos têm sido desenvolvidos. A relação entre fractal e codificadores baseados em transformada foi investigada em [79]-[83]. Davis [81][84][85] apresentou uma aproximação que usa elementos de ambos os tipos de compressão: fractal e *wavelet*. Em [82] argumenta-se que o mapeamento contrativo fractal poderia ser considerado como uma operação de predição do domínio *wavelet*. Em [83] os autores fizeram o *matching domain-range* no domínio *wavelet* para obter uma compressão mais alta.

Li e Kuo [86] usaram o mapeamento contrativo fractal para predizer os coeficientes *wavelets* entre escalas e então codificaram o resíduo de predição com um codificador de plano de bits. Esse

procedimento é diferente de [80] e [81] e de outros codificadores fractais convencionais, onde a imagem é codificada inteira por predição fractal.

O número bastante restrito dessas publicações sobre Codificadores Híbridos Fractal-*Wavelet* e o fato de seus autores não explicitarem todos os parâmetros utilizados (dada a abrangência das duas técnicas) torna difícil a comparação via implementação desses codificadores, uma vez que existe quase uma impossibilidade de reproduzir seus resultados. Em contrapartida, os codificadores híbridos se tornam um vasto campo para a pesquisa.

Codificação Entrópica Fractal

Com exceção da alocação de bits que Fisher utilizou para a *quadtree* (apresentada na Seção 11.1), não é aplicado nenhum tipo de compressão *lossless* (sem perdas) à informação fractal codificada, mesmo nos trabalhos recentes. Por esse motivo e por fugir ao escopo e abrangência da tese, optou-se por não implementar nenhuma codificação entrópica neste trabalho além da de Fisher.

Simulações revelam que a distribuição de probabilidade típica da estrutura da informação enviada é bastante semelhante à distribuição de probabilidade uniforme, com exceção do parâmetro o_i. Assim, a codificação entrópica somente traria algum ganho no caso desse parâmetro. Logo, nesta pesquisa o gasto de bits para o_i foi estimado pela entropia de primeira ordem e acrescido aos demais gastos de bits.

11.4 – Comentários

Este capítulo apresentou os dois sistemas-base fractais de comparação: o Codificador de Fisher [3] e o Codificador de Cardinal [4]. A bibliografia referente a outros codificadores acelerados e híbridos é também apresentada de forma a situar o leitor nas diversas possibilidades. O capítulo seguinte apresenta finalmente o codificador híbrido proposto e seus princípios de funcionamento.

Este capítulo encerra a abordagem da Teoria Fractal, nas quais foram fundamentados os conceitos necessários à apresentação no Capitulo 12 de: 1) a parcela fractal do Codificador Híbrido Fractal-*Wavelet* proposto neste trabalho; 2) o Sistema-Base 2 (Codificador de Fisher, referenciado por "Fractal Acelerado") e 3) o Sistema-Base 3 (Codificador de Cardinal, referenciado simplesmente por "Cardinal").

Capítulo 12

Codificador Híbrido Fractal-*Wavelet* Proposto e Sistemas-Base de Comparação

O objetivo deste trabalho é propor um método de aceleração para a exaustiva compressão fractal de imagens estáticas, através da aplicação da TWD e suas propriedades sem que haja detrimento do PSNR ou da qualidade visual. Logo, este trabalho apresenta um novo Codificador Híbrido Fractal-*Wavelet*, que aplica a compressão fractal acelerada à imagens estáticas decompostas pela Transformada *Wavelet*, explorando a direcionalidade das subimagens-*wavelet*, constatada por Shapiro [2].

Através deste esquema, para diversas imagens e *bitrates*, obteve-se uma melhora visual corroborada pelos valores de PSNR e um aumento na velocidade de processamento fractal pura em cerca de 80% para uma mesma taxa de bits, atingindo o objetivo proposto.

O objetivo deste capítulo é apresentar o Codificador Híbrido proposto e sumarizar os Sistemas-Base 1, 2 e 3, com os quais ele será comparado. Trata-se de um capítulo compacto, uma vez que toda a base teórica necessária à compreensão dos Sistema-Base e Proposto foi cuidadosamente apresentada ao longo dos capítulos anteriores. Optou-se por esta estrutura de forma a isolar a teoria já conhecida das contribuições originais propostas por esta pesquisa. Os detalhes e parâmetros de implementação propriamente ditos são também apresentados.

Todos os sistemas e o Codificador Proposto foram todos implementados em *Matlab* 6.1. Os resultados experimentais encontram-se no Capítulo 13.

12.1. – Sistema-Base 1: Codificador *Wavelet* Puro ("TWD+SPIHT")

O Sistema-Base 1 (técnica de compressão *lossy wavelet*) pode ser representado esquematicamente na forma da Fig. 12.1, onde podem ser observadas a etapa de codificação e decodificação. A TWD adotada para ambos os sistemas (base e codificador híbrido proposto) foi a *wavelet* biortogonal *spline* 9-7. Utilizou-se 3 níveis de decomposição.

De forma geral, os únicos componentes não inversíveis da Fig 12.1 são as etapas de quantização/dequantização que, no caso de esquemas que adotam o algoritmo SPIHT, encontram-se internas a ele. Esse componente é, portanto, responsável pelas perdas do sistema. Dentre os métodos de compressão *lossy* para coeficientes-*wavelet*, existem quatro algoritmos principais recentes e que

obtém os menores erros por taxa de compressão, bem como os resultados visuais mais expressivos já reportados. Tratam-se dos algoritmos EZW (*Embedded Zerotree Wavelet*), SPIHT (*Set Partitioning in Hierarchical Trees*), WDR (*Wavelet Difference Reduction*) e ASWDR (*Adaptatively Scanned Wavelet Difference Reduction*) [41].

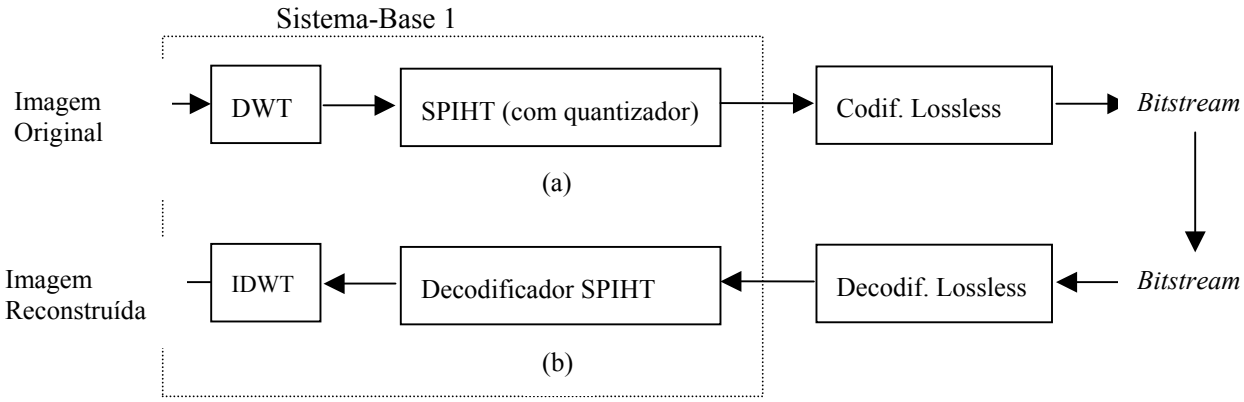


Fig. 12.1. Esquema do Sistema-Base 1 ("TWD +SPIHT"): (a) codificação; (b) decodificação.

O algoritmo SPIHT, escolhido para implementação desta pesquisa e núcleo do Sistema-Base 1 é apontado por Rao e Yip como sendo provavelmente o algoritmo de compressão de coeficientes-*wavelet* mais largamente utilizado, servindo de padrão básico de comparação para algoritmos subseqüentes [41]. Nele, é utilizado um particionamento melhorado das árvores de coeficientes de forma a manter os coeficientes-*wavelet* insignificantes melhor agrupados nos subconjuntos. As decisões de particionamento (binárias) 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 [41].

O Sistema-Base 1 foi implementado segundo o esquema da Fig. 12.1, seguindo rigorosamente a seção 7.4. Foi implementada a varredura apresentada em Yip e Rao [41] que privilegia as características de cada tipo de subimagem (horizontal, diagonal ou vertical), de forma a facilitar uma eventual compressão *lossless* subseqüente. Não foi executada codificação aritmética.

12.2 – Sistema-Base 2: Codificador de Fisher ("Fractal Acelerado") [3]

Para comparar o codificador proposto com técnicas fractais "puras", foram escolhidos dois codificadores. O primeiro, o Codificador Fractal com Acelerador (Fisher) [3], foi escolhido como Sistema-Base 2, por ser um dos codificadores mais implementados na literatura científica atual.

O Sistema-Base 2 (técnica de compressão *lossy* fractal pura) pode ser representado esquematicamente na forma da Fig. 12.2, onde podem ser observadas a etapa de codificação e decodificação. Novamente, os componentes da Fig. 12.2 não-inversíveis (portanto, responsáveis pelas perdas do sistema) são as etapas de "quantização e dequantização inerentes ao processo fractal.

No Capítulo 10, foi visto que as técnicas fractais aplicadas à compressão de imagens consistem fundamentalmente na busca por auto-similaridades, baseando-se no conceito de que a imagem é formada por cópias propriamente transformadas de partes de si mesma. Assim, à cada bloco da imagem de entrada (*domain-block*) são aplicadas subamostragem, filtragem de média e transformações-afins ω_i . Os resultados das transformações são comparados à cada bloco a ser codificado (*range-block*).

Uma vez encontrado o *matching* segundo um critério pré-determinado, são enviados basicamente ao decodificador os coeficientes da transformação afim responsável pela formação de cada *range-block* e o endereço do *domain-block* escolhido. O conjunto de todos os *domain blocks* é chamado de *Domain pool*.

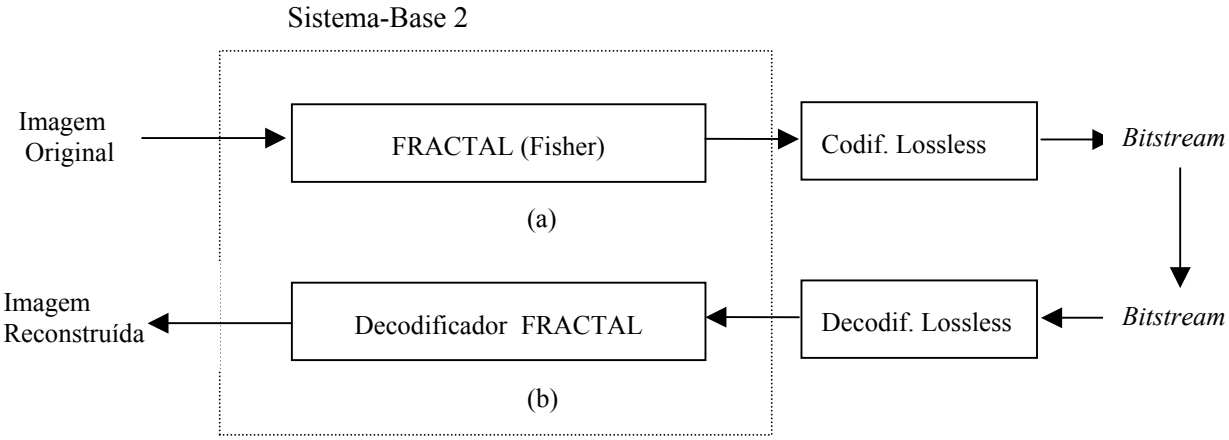


Fig. 12.2. Esquema do Sistema-Base 2 ("FRACTAL ACELERADO"): (a) codificação; (b) decodificação.

O Codificador fractal acelerado de Fisher, suas características e funcionamento foram abordados no Capítulo 11. Basicamente se trata de um esquema de classificação de domínios de forma a reduzir a *Domain pool* [3]. O Sistema-Base 2 foi implementado segundo [3], com os seguintes parâmetros utilizados por Fisher:

1 – Métrica rms ou euclidiana para comparar os pares *domain-range*:

$$d\left(f(x,y),g(x,y)\right)=\frac{1}{MN}\left(\sum_{x=1}^{N}\sum_{y=1}^{M}\left\{f(x,y)-g(x,y)\right\}^2\right)^{\frac{1}{2}} \tag{12.1}$$

onde MxN são as dimensões dos blocos.

2 – Nível máximo e mínimo de *quadtree* respectivamente 7 e 3 (normalmente utilizado para imagens 512x512 que foram as utilizadas nos experimentos desta pesquisa).

3 – Tipo de *Domain Pool*: D2 com *overlap* de 50%.

4 – Acelerador de Fisher: classificação de média e variância (superclasse com subclasse). Observando que, caso aplicando-se ambas, não restarem blocos-domínio possíveis, utiliza-se somente a superclasse.

5 – Máximo fator de contraste: 1,2.

6 – Quantização do par $[S_i, O_i] = [5, 7]$ bits.

7 – Sem codificação *lossless*. O número de bits foi estimado pela entropia de primeira ordem.

12.3 – Sistema-Base 3: Codificador de Cardinal ("Cardinal") [4]

O segundo codificador fractal "puro"utilizado como base de comparação, o Codificador Fractal de Cardinal [4], foi escolhido por ser uma referência bastante atualizada e requisitada como um bom parâmetro de comparação pela revista científica *IEEE Transactions on Image Processing* em novembro de 2004.

A codificação e decodificação do Sistema-Base 3 (técnica de compressão *lossy* fractal pura) é ilustrado na Fig. 12.3. Novamente, os componentes não-inversíveis são as etapas embutidas no algoritmo fractal em questão.

Cardinal [4] faz uso de resultados de outros autores, reduzindo o processo de procura pelas transformações afins ao problema geométrico clássico de busca pelos vizinhos mais próximos num espaço euclidiano. Para cada um dos blocos (*domain-blocks* e *range-blocks*) é calculado um vetor de características (também chamado de "*key*"). O espaço de procura contendo todas as *keys* é então particionado recursivamente.

Esse particionamento é feito até que um número pequeno de *domain-keys* seja deixado em cada subespaço. Cada *range-block* (correspondendo a uma *range-key* no subespaço) é então

comparado a cada um dos *domain-blocks* (correspondendo às *domain-keys* remanescentes). A comparação que gere menor rms fornecerá o *domain-block*-equivalente, semelhantemente a Fisher. A informação de particionamento do espaço de procura não precisa ser enviada.

Baseando-se nessa partição geométrica do espaço de características, Cardinal conseguiu uma aceleração excelente na compressão sem perda de qualidade, quanto comparado a diversos resultados anteriores. O codificador fractal de Cardinal e suas características foram abordadas no Capítulo 11.

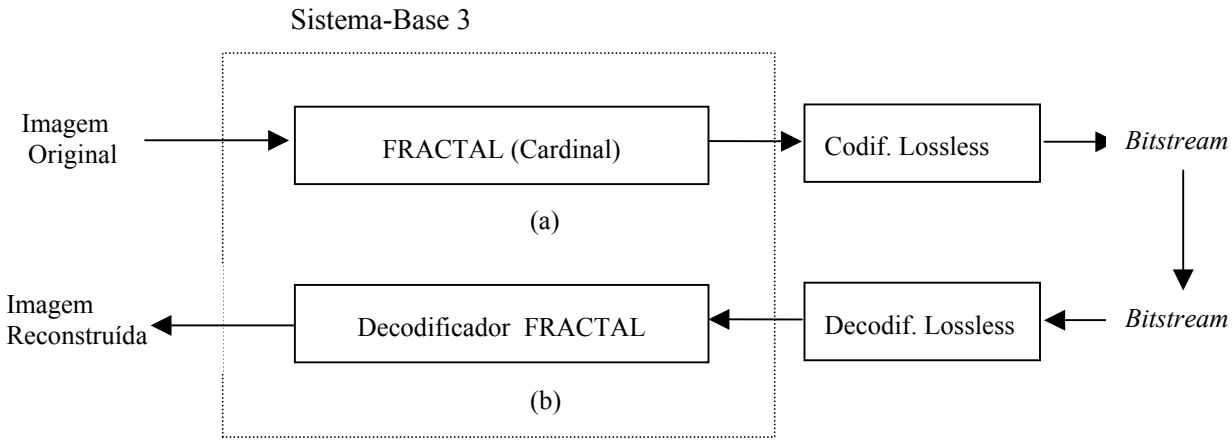


Fig. 12.3. Esquema do Sistema-Base 3 (“CARDINAL”): (a) – codificação; (b) – decodificação.

Os resultados do Codificador Proposto foram comparados diretamente com os resultados apresentados em [4]. Logo, o Sistema-Base 3 não foi implementado para os testes. Contudo, os parâmetros de simulação constantes em [4] são:

1 – Métrica rms ou euclidiana para comparar os pares *domain-range*, segundo (12.1):
2 – Nível máximo e mínimo de *quadtree* respectivamente 7 e 5.
3 – Tipo de *Domain Pool*: D1 com espaçamento da grade igual a 2 *pixels*.
4 – Classificação usando o operador de projeção de Saupe e divisão por hiperplanos. Número máximo de *domain-keys* por célula:300.
5 – Máximo fator de contraste: 1,0.
6 – Não usou transformações geométricas (para cada comparação *domain-range* é feito um único cálculo de s_i e o_i).
7 – Quantização do par $[S_i, O_i] = [5, 7]$ bits.
8 – Sem codificação *lossless*.

12.4 – Codificador Híbrido proposto ("Fractal-*Wavelet*")

O Codificador Híbrido Proposto pode ser *genericamente* representado segundo o esquema da Fig. 12.4. Contudo, a combinação das vantagens das técnicas fractal e *wavelet* e de como elas são exploradas será revista na seqüência. Conforme já colocado, a TWD adotada foi a *wavelet* biortogonal *spline* 9-7 e o responsável pelas perdas do sistema encontra-se embutido no algoritmo fractal.

Sabe-se que a TWD decompõe a imagem original em *3k* subimagens de detalhes e *1* subimagem de aproximação, conforme pode ser observado na Fig. 12.5, onde *k* é o nível de decomposição *wavelet* desejado. Logo, a TWD decompõe a imagem original em um conjunto de subimagens com diferentes resoluções correspondentes a diferentes bandas de freqüências, fornecendo uma representação da estrutura espacial da imagem em diferentes escalas. Claramente, as características espaciais de uma subimagem particular em diferentes resoluções são diferentes, mas altamente correlatas uma vez que, na verdade, especificam a mesma estrutura espacial em diferentes escalas [6][9].

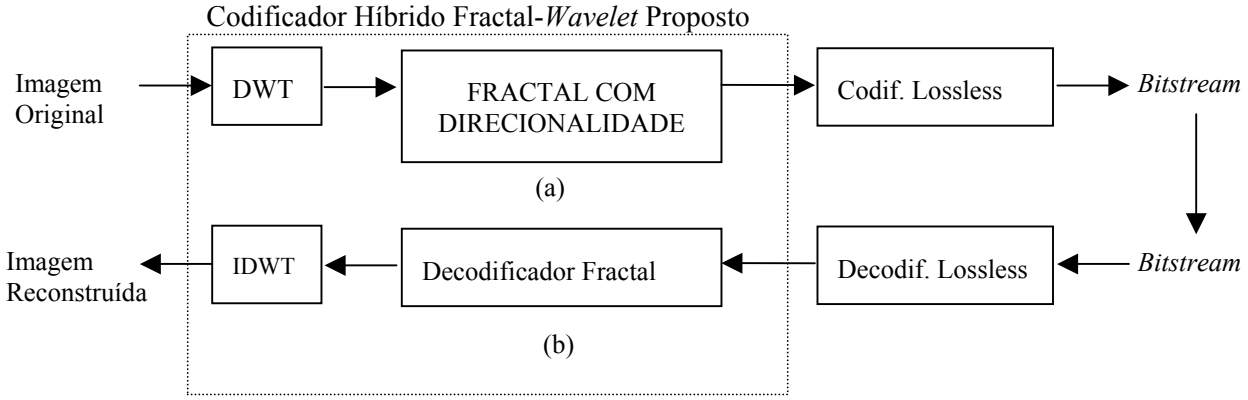
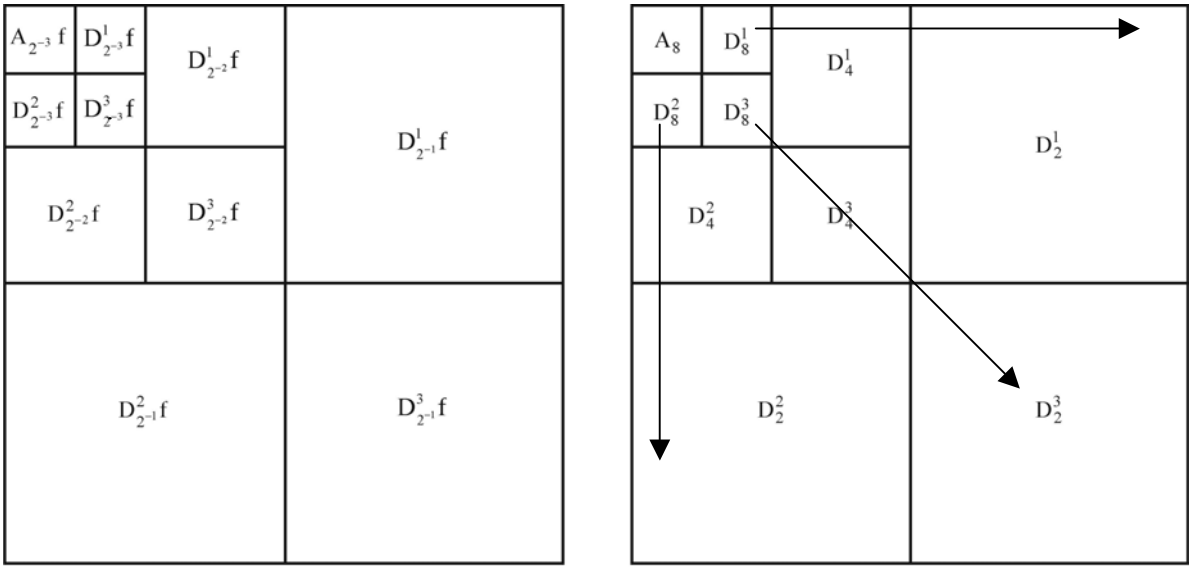


Fig. 12.4. Esquema do Codificador Híbrido Proposto ("FRACTAL-*WAVELET*"): (a) codificação; (b) decodificação.

Essa característica de correlação entre as subimagens foi explorada por Shapiro em seu esquema EZW de codificação zerotree [2] e estendida para vídeo através de esquemas de estimação de movimento multiresolução apresentados por diversos autores como em [87]-[89].

Assim, as subbandas verticais de diferentes escalas apresentam características de alta correlação espacial entre si, ocorrendo o mesmo para as subbandas horizontais e diagonais. Essa característica será referenciada neste trabalho como "*direcionalidade*" (Fig. 12.5.b).



(a) (b)

Fig. 12.5. Imagens resultantes da decomposição *wavelet* ortogonal 2D para 3 níveis de decomposição: (a) notação detalhada; (b) notação simplificada e direcionalidade.

Já com relação à codificação fractal, o objetivo é o encontro de auto-similaridades de trechos da imagem dentro de si mesma em escala maior, de forma a encontrar o que pode ser chamado de *par bloco-domínio-imagem*. Assim, ao invés de promover o envio do bloco, somente são enviados os coeficientes da transformação, o que se reflete em economia de transmissão.

A idéia central desta pesquisa reside então na exploração dessas características de direcionalidade *wavelet* e poderosa compressão fractal. Primeiramente, consideremos uma imagem original decomposta pela TWD e focalize-se a atenção na subbanda vertical D^1_8. Aplicando-se a codificação fractal (de Fisher [3], por exemplo) nessa subimagem, determinam-se os pares domínio-imagem para seus blocos.

Aplicando o conceito de direcionalidade para tirar proveito da altíssima correlação entre as subbandas de mesmo tipo (no caso, verticais) em diferentes escalas, pode-se dizer, observando a Fig. 12.6, que os pares domínio-imagem de D^1_8 serão muito próximos aos pares domínio-imagem de D^1_4 e de D^1_2, bastando adaptar o tamanho dos blocos ao seu nível da pirâmide *wavelet*.

Assim, os pares domínio-imagem de um determinado tipo de subimagem acabam sendo preditos pelos pares domínio-imagem da subimagem de menor escala do mesmo tipo, sem necessidade de procura em *Domain Pool*. Nessa idéia reside a inovação central proposta pela

pesquisa. Claramente, pode ser feito também um refinamento $\Delta V(x,y)$ após a predição inicial, conforme ilustrado na Fig. 12.6.

A questão da velocidade computacional se apresenta como o principal motivo de hesitação da adoção de esquemas fractais. A exploração da direcionalidade *wavelet* substitui o uso de *Domain Pools* para as camadas de maior resolução, solucionando essa questão.

A codificação *lossless* (sem perdas) representada nos 4 esquemas (3 Sistemas-Base de comparação e Proposto) não foi implementada por questão de isolamento de variáveis, de forma a facilitar a interpretação dos resultados.

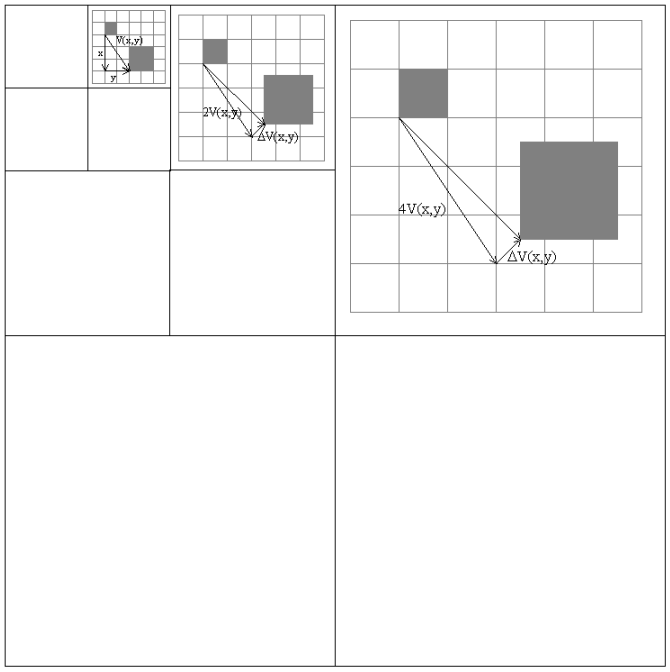


Fig. 12.6. Predição dos pares domínio-imagem das subimagens verticais pelos pares domínio-imagem da subimagem de menor escala do mesmo tipo.

Melhorias Adicionais

Esquemas fractais comumente são aplicados sobre a imagem original, como ocorre nos Sistemas Base 2 e 3. Porém, no Codificador Híbrido, técnicas fractais são aplicadas à subimagens de coeficientes-*wavelet*, que apresentam características tais como tendência de concentração de energia rumo ao canto superior esquerdo da decomposição e baixíssima energia nas subbandas de detalhes. Dessa forma, não convém utilizar simplesmente o mesmo padrão de quantização em ambos os sistemas. Há a necessidade de adaptação.

Para essa adaptação, foram feitas algumas melhorias adicionais no codificador, priorizando a tendência de concentração de energia citada e evitando o desperdício de bits nas subbandas de detalhe, aumentando consecutivamente as taxas de compressão de operação do sistema.

- Camada de detalhes 3 (A_8, D_8^1, V_8^1, H_8^1) da TWD:

Esta camada é tratada de forma exclusiva por concentrar a maior parcela da energia da imagem. Para ela é feita a codificação fractal mais detalhada através dos parâmetros de quantização do par [S_i , O_i] setados em [7,9] para a passa-baixas e [3,5] para as demais. São utilizados blocos de dimensões 2x2.

- Demais Camadas:

Para as demais camadas foi elaborado um controle de qualidade (e por conseguinte, de *bits*) através da aplicação de um *threshold T* , semelhante à filosofia básica do SPIHT. Comparados a T, os coeficientes do bloco são classificados em significativos ou não. Caso todos os coeficientes do bloco sejam não-significativos, serão quantizados segundo o par [2,2] e processados normalmente.

Caso haja pelo menos um coeficiente significativo, o bloco será particionado em *quadtree*, até formar subblocos de dimensões 2x2, de forma a reduzir o número de coeficientes a serem refinados e elevar a precisão de sua codificação. Para esses *Range-Blocks* significativos 2x2, serão feitos: um refinamento na procura pelo par range-domain (através de uma janela de procura com *overlap* que parte do ponto inicial previsto pela direcionalidade) e uma melhor quantização ([3,5] à semelhança da camada 3).

Os *range-blocks* não-significativos 2x2 serão quantizados segundo o par [2,2] e processados normalmente. A Fig. 12.7 apresenta o modelo esquemático da situação de um *range-block* que possui coeficiente significativo acima descrita.

Na Fig. 12.7, o bloco 4x4 em D_4^1 possui ao menos um coeficiente significativo sendo, portanto, particionado formando subblocos 2x2. Para estes subblocos 2x2 será feito o refinamento na procura pelo par *range-domain*. Assim, partindo do ponto inicial previsto pela direcionalidade (no caso, o ponto A), são feitas novas comparações de *matching* entre os *range*-subblocos 2x2 e os *domain*-subblocos 4x4 (estes 4x4 estão dentro da janela de procura de dimensões 8x8). Uma vez determinado o melhor *matching*, será utilizada a citada quantização [3,5].

O valor de T é parâmetro de entrada e funciona como um refinamento na codificação, na medida em que proporciona que mais blocos sejam melhor codificados. Exerce, dessa forma,

função de controle de qualidade e, por conseguinte, de bits.

- Escape:

A codificação fractal apresenta exceções de codificação localizadas, ou seja, por vezes não encontra auto-similaridade. Para essas exceções, o melhor *matching* do par domínio-imagem pode apresentar erro considerável. Para eliminar essa questão, também foi implementado um mecanismo de escape. Caso o erro do *matching* seja superior a um determinado valor, o bloco é transmitido sem codificação. No caso, o escape é setado em 5% da máxima amplitude em módulo dos coeficientes, para a passa-baixas; e 5% da máxima amplitude em módulo de todos os coeficientes para as demais subimagens. Esse valor foi determinado através de diversos testes, tendo apresentado ótimo resultado para diversas imagens.

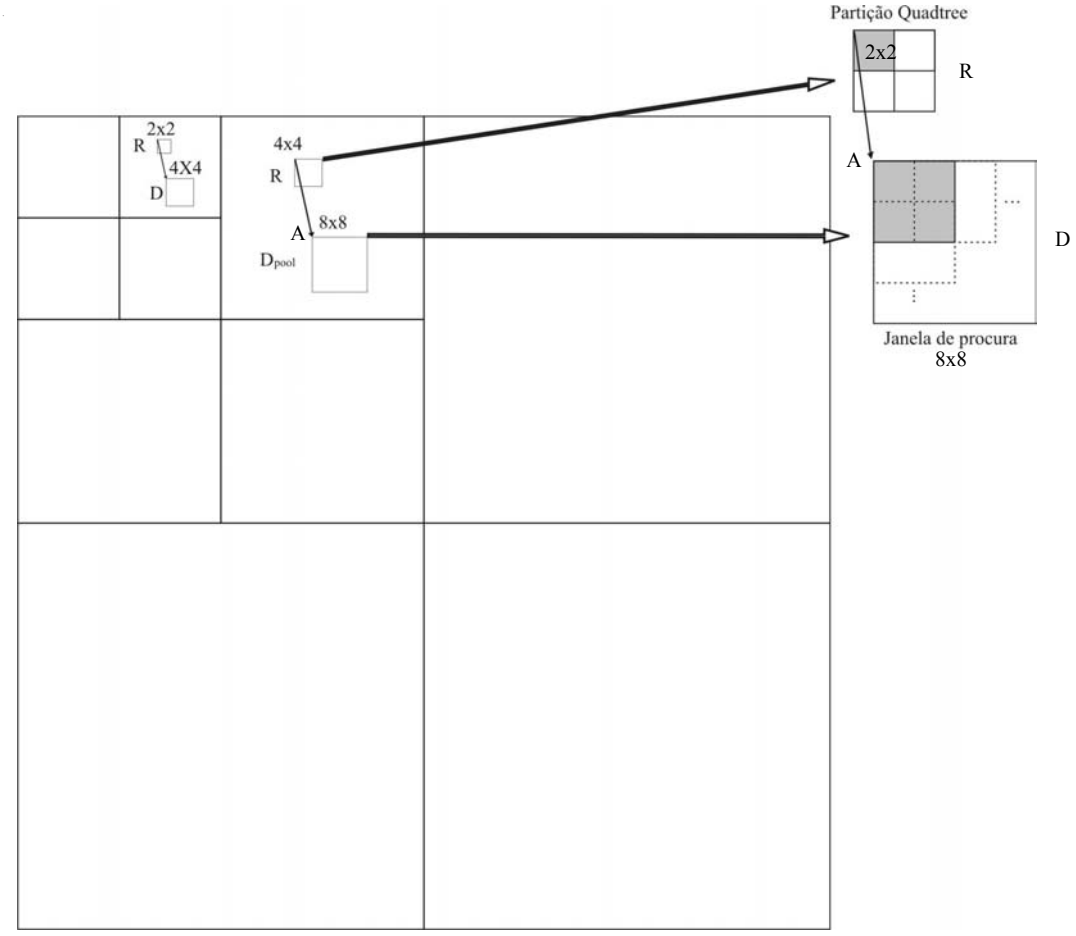


Fig. 12.7. Modelo esquemático do processamento de um *range-block* 4x4 que possui coeficiente significativo.

Deve-se ressaltar que outros valores de quantização [S_i , O_i] deste processo como um todo foram testados. O que pudemos observar foi um ótimo desempenho, desde que se mantenha a

priorização da concentração de energia rumo ao canto superior esquerdo da decomposição *wavelet*.

As comparações e resultados completos encontram-se no capítulo seguinte. O Codificador Proposto ("Fractal-*Wavelet*") foi implementado com os seguintes parâmetros:

1 – Métrica rms ou euclidiana para comparar os pares domínio-imagem, segundo (12.1).
2 – *Threshold T* adotado como parâmetro de entrada (controle de qualidade).
3 – Tipo de *Domain Pool*: D2 com *overlap* de 50%.
4 – Acelerador de Fisher (superclasse com subclasse) aplicado somente nas subimagens de último nível *wavelet* (nível 3 de decomposição, passa-baixas inclusive). Não aplicado às demais em virtude da proposta (aplicar direcionalidade para as demais subimagens, evitando novas procuras fractais através da eliminação de *Domain Pools*). Observando que, aplicando-se superclasse com subclasse, case não restem *domain-blocks* possíveis, utiliza-se somente a superclasse.
5 – Predição dos pares *domain-range* de um determinado tipo de subimagem através dos pares domínio-imagem da subimagem de menor escala do mesmo tipo. Predição direcional sem refinamento.
6 – Refinamento (janela de procura) para blocos significativos.
7 – Máximo fator de contraste: 1,2 (segundo Fisher)
8 – Quantização do par $[S_i, O_i] = [7, 9]$ bits para as subimagens passa-baixas; [3, 5] para o nível 3 de decomposição e para todos os blocos dados como significativos segundo T (após partição *quadtree*); e [2,2] para blocos não-significativos (antes ou após partição *quadtree*, dependendo do caso).
9 – Mecanismo de escape setado em 5% da máxima amplitude em módulo dos coeficientes, para a passa-baixas; e 5% da máxima amplitude em módulo de todos os coeficientes para as demais subimagens. Esse mecanismo controla eventuais erros localizados de *matching* fractal.
10 – Sem codificação *lossless*. O número de bits foi estimado pela entropia de primeira ordem.

12.5 – Comentários

Foram detalhados neste capítulo os princípios de funcionamento do Codificador Proposto ("FRACTAL-*WAVELET*"), cuja idéia central é a exploração conjunta da codificação fractal com a direcionalidade *wavelet*, procedimento que elimina a necessidade de *Domain Pools* para grande parte das subimagens e que tem como conseqüência direta o aumento de velocidade de codificação.

Também foram sumarizados e identificados os Sistemas-Base de Comparação que foram utilizados nos testes:

1) Sistema-Base 1: Codificador *Wavelet* Puro (“TWD+SPIHT”);
2) Sistema-Base 2: Codificador de Fisher (“Fractal Acelerado”) [3];
3) Sistema-Base 3: Codificador de Cardinal (“Cardinal”) [4] .

O Codificador/Decodificador Híbrido Fractal-*Wavelet* completo foi apresentado, contando com melhorias adicionais na idéia básica: um esquema de controle de qualidade e um mecanismo de escape (prevenindo exceções de ausência de auto-similaridade).

Os parâmetros de simulação tanto dos Sistemas-Base como do Codificador Proposto foram também enumerados. O capítulo seguinte apresentará os resultados experimentais das simulações e as respectivas análises realizadas durante o desenvolvimento do trabalho.

Capítulo 13

Resultados Experimentais

Neste capítulo são apresentados os resultados experimentais obtidos pelo Codificador Híbrido Fractal-*Wavelet* proposto para acelerar a velocidade da compressão fractal de imagens estáticas, sem que haja detrimento do PSNR ou da qualidade visual. Os resultados obtidos são analisados com base no tempo de processamento (preocupação primordial ao se trabalhar com fractal e *wavelets*), na qualidade visual das imagens recuperadas, na medida de PSNR (*Peak Signal-to Noise Ratio*) e no número de bits por *pixel* (*bitrate*) utilizado na codificação das seqüências.

Conforme será apresentado ao longo deste capítulo, o codificador proposto apresentou, para diversas imagens e *bitrates*, uma melhora visual (corroborada pelos valores de PSNR) e um aumento na velocidade de processamento fractal pura em cerca de 80% para uma mesma taxa de bits, tornando viável a exploração da ferramenta de compressão conjunta. Nenhum efeito de bloco (característico de procedimentos fractais puros) resultou do processo híbrido. Os detalhes e as características de transmissão progressiva *wavelet* foram também preservados com o esquema proposto. O aumento na complexidade do codificador pode ser considerado de pequeno porte.

13.1 – Introdução e Observações Sobre as Simulações

O PSNR entre a imagem original (F_o) e a imagem reconstruída (F_R), ambas de dimensões *WxH* com 8bpp pode ser expresso como função do MSE (*Mean Square Error*):

$$PSNR\left(F_o, F_R\right) = 10 \cdot log\left(\frac{255^2}{MSE\left(F_o, F_R\right)}\right), \text{ onde} \tag{13.1}$$

$$MSE\left(F_o, F_R\right) = \frac{1}{W \cdot H} \sum_{x=0}^{H-1} \sum_{y=0}^{W-1} \left(F_o(x,y) - F_R(x,y)\right)^2 \tag{13.2}$$

No total dos experimentos foram utilizadas 8 imagens 512x512 do conjunto-padrão em escala de cinza disponível no *website* Waterloo Bragzone[1]. Essas imagens são de domínio público e extensamente utilizadas na literatura científica do processamento digital de imagens.

Os codificadores testados (Sistemas-Base e Codificador Proposto) foram implementados em Matlab 6.1, que também foi o ambiente de programação para a visualização de imagens e construção de gráficos. Os tempos de processamento englobam os valores de codificação e decodificação.

[1] http://links.uwaterloo.ca

Embora sistemas de compressão de imagens completos empreguem codificação entrópica e pós-processamento nas imagens decodificadas para melhorar os resultados, nesta pesquisa esses procedimentos não foram adotados, uma vez que o objetivo era enfocar especificamente as implementações fractais e *wavelet*. Essa opção permitiu o isolamento de variáveis e uma melhor análise do núcleo dos processos. Entretanto, para a estimação do *bitrate* das imagens codificadas, foi utilizado o cálculo de entropia de primeira ordem.

O Codificador Híbrido Proposto será referenciado por ("FRACTAL-*WAVELET*"). Os Sistemas-Base para Comparação com o Codificador Híbrido Proposto neste trabalho serão referenciados conforme a prévia abordagem teórica (Capitulo 12, em especial), ou seja:

1) Sistema-Base 1: Codificador *Wavelet* Puro ("TWD+SPIHT");
2) Sistema-Base 2: Codificador de Fisher ("FRACTAL ACELERADO") [3];
3) Sistema-Base 3: Codificador de Cardinal ("CARDINAL") [4] .

Divide-se a apresentação dos resultados e comentários em 3 seções: 1) seção 13.2, relativa aos resultados com análise subjetiva; 2) seção 13.3, relativa aos resultados estendidos; 3) seção 13.4, contendo o resumo e comentário das conclusões.

13.2 – Resultados com Análise Subjetiva

A Fig. 13.1 mostra as imagens-teste originais *Lena, Peppers e Mandrill*, extremamente utilizadas na literatura científica. São apresentadas suas dimensões e algumas de suas regiões serão ampliadas para tornar clara a qualidade da imagem após os processamentos pelos Sistemas-Base e Proposto, permitindo as comparações.

(a) (b) (c)

Fig. 13.1 – Imagens originais (512x512 *pixels*). (a) *Lena*; (b) *Peppers*; (c) *Mandrill*.

As Figs. 13.2 e 13.3 mostram a imagem *Lena* original e as imagens reconstruídas pelos diversos métodos a 0,52 bpp. As medidas de PSNR encontram-se indicadas nas figuras em questão.

(a)

(b) PSNR = 32,97dB

Fig. 13.2 - Imagem *Lena* 512x512, 0,52bpp : (a) ORIGINAL; (b) reconstruída TWD+ SPIHT.

(a) PSNR = 32,68 dB

(b) PSNR = 33,60 dB

Fig. 13.3 - Imagem *Lena* 512x512 reconstruída, 0,52bpp:
(a) FRACTAL ACELERADO; (b) FRACTAL-*WAVELET*.

Do ponto de vista da qualidade visual pode ser observado da Fig. 13.2 que o sistema TWD+SPIHT apresenta realmente um ótimo desempenho, somente gerando distorções de brilho devido ao seu esquema de quantização, o que pode ser observado em especial ao longo dos traços do rosto e penacho do chapéu de *Lena*. Nessa taxa de compressão (0,52bpp) também observou-se perda de textura em regiões homogêneas, como a bochecha de *Lena*.

O sistema FRACTAL ACELERADO da Fig. 13.3a apresenta grande perda de detalhamento e artefatos (efeitos de bloco em especial na região dos olhos, nariz e ombros de *Lena*). O Codificador Híbrido FRACTAL-*WAVELET* da Fig. 13.3b apresenta qualidade de imagem ligeiramente superior ao TWD+SPIHT e superior ao Fractal Acelerado, com a vantagem de não causar a distorção de brilho do primeiro, nem os artefatos e perdas do segundo. Contudo, há um leve aumento no efeito de *ringing*.

Do ponto de vista de PSNR das imagens reconstruídas, cujos valores encontram-se nas respectivas figuras, o Sistema Híbrido FRACTAL-*WAVELET* apresenta PSNR superior aos demais. Embora seja conhecido o fato de que o PSNR nem sempre reflete a qualidade visual com precisão [41], essa medida apresentou-se coerente com a análise visual.

Quanto ao aspecto da velocidade computacional, extremamente importante, tem-se que o tempo de processamento para essa imagem foi: 3’28’’ para o TWD+SPIHT; 26’15’’ para o FRACTAL ACELERADO e somente 6’02’’ para FRACTAL-*WAVELET*, demonstrando que o objetivo de minimizar o gasto computacional fractal é plenamente alcançado através da aplicação da direcionalidade da TWD. Contudo, o melhor resultado nesse aspecto é obtido com o TWD+SPIHT. Fica evidenciado um fator importante: para um mesmo número de bits e qualidade visual melhor, FRACTAL-*WAVELET* reduz o tempo de processamento fractal em 77%.

Nas Figs. 13.4 e 13.5 são apresentadas as ampliações da imagem *Lena* de forma a permitir as comparações necessárias entre os sistemas de codificação, enfocando regiões de detalhes. Pode ser observada a citada distorção de brilho nas ampliações referentes ao TWD+SPIHT, em especial em torno dos traços do rosto, ombro e penacho do chapéu de *Lena*. FRACTAL ACELERADO apresenta elevadíssima perda de detalhamento e efeito de bloco considerável. No detalhamento pode-se observar mais claramente que FRACTAL-*WAVELET* apresenta qualidade superior aos demais (evita a distorção de brilho e a blocagem). As Tabelas XIII.1-XIII.3 no fim da seção apresentarão um número maior de comparações.

Fig. 13.4 – Ampliações da imagem *Lena* reconstruída a 0,52bpp:
(Lado esquerdo) ORIGINAL; (Lado direito) TWD+SPIHT.

Fig. 13.5 – Ampliações da imagem *Lena* reconstruída a 0,52bpp:
(Lado esquerdo) FRACTAL ACELERADO; (Lado direito) FRACTAL-*WAVELET*.

Nas Figs. 13.6 e 13.7 encontram-se a imagem *Peppers* original e as imagens reconstruídas pelos diversos métodos a 0,52bpp. As medidas de PSNR encontram-se indicadas.

(a)

(b) PSNR = 32,65 dB.

Fig. 13.6 - Imagem *Peppers* 512x512, 0,52bpp : (a) ORIGINAL; (b) reconstruída TWD+ SPIHT.

(a) PSNR = 33,86 dB

(b) PSNR = 33,80dB

Fig. 13.7 - Imagem *Peppers* 512x512 reconstruída, 0,52bpp :
(a) FRACTAL ACELERADO; (b) FRACTAL-*WAVELET*.

Novamente analisando a qualidade visual, pode ser observado da Fig. 13.6 que o sistema TWD+SPIHT apresenta novamente bom desempenho, mas ainda gera distorções de brilho e perda de textura, o que pode ser observado em especial nas áreas planas (por exemplo, a pimenta vertical à esquerda) de *Peppers*.

O sistema FRACTAL ACELERADO da Fig. 13.7a apresenta novamente excessiva perda de detalhamento e artefatos de blocagem ao longo de toda a imagem. O sistema híbrido FRACTAL-*WAVELET* da Fig. 13.7b, apresenta qualidade de imagem superior ao TWD+SPIHT e ao Fractal Acelerado, novamente não causando a distorção de brilho e perda de textura do primeiro, nem a blocagem do segundo.

Analisando o PSNR das imagens reconstruídas, cujos valores encontram-se nas respectivas figuras, o sistema híbrido FRACTAL-*WAVELET* apresenta PSNR equivalente ao do sistema FRACTAL ACELERADO e ligeiramente superior ao TWD+SPIHT.

Quanto à velocidade computacional, tem-se que o tempo de processamento para esta imagem foi: 3’47’’ para o TWD+SPIHT; 30’06’’ para o FRACTAL ACELERADO e somente 6’20’’ para FRACTAL-*WAVELET*, confirmando a minimização do gasto computacional fractal através da aplicação da direcionalidade da TWD. Novamente o melhor resultado nesse aspecto é obtido com o TWD+SPIHT.

Nas Figs. 13.8 e 13.9 são apresentadas as ampliações referentes à Imagem *Peppers*, enfocando regiões de detalhes. Pode ser observada novamente a citada distorção de brilho e perda de textura do TWD+SPIHT, e a elevadíssima perda de detalhamento e artefatos de blocagem graves no sistema-base FRACTAL ACELERADO.

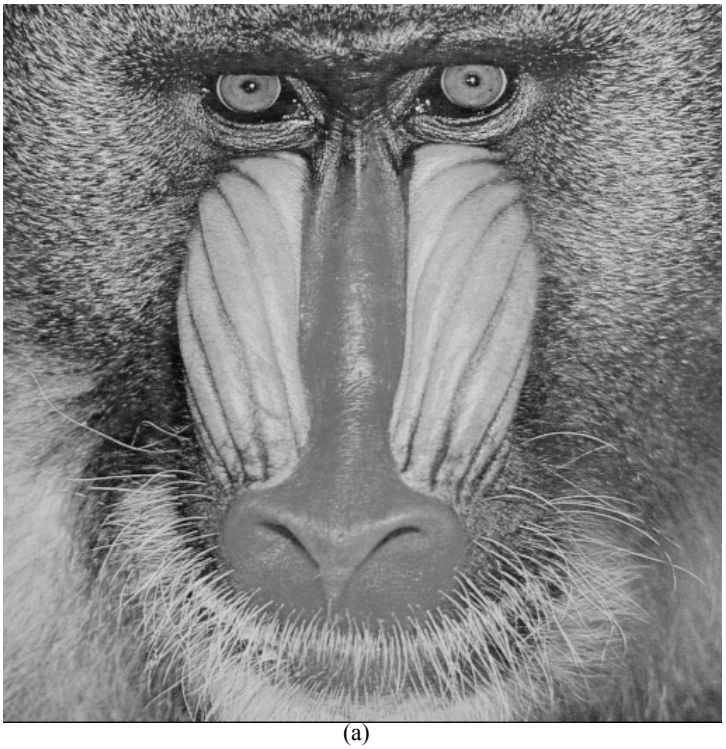
No detalhamento pode-se observar novamente que FRACTAL-*WAVELET* apresenta qualidade subjetiva superior a ambos.

As Figs. 13.10 e 13.11 mostram a imagem *Mandrill* original e as imagens reconstruídas pelos diversos métodos a 0,52 bpp. Nas Figs. 13.12 e 13.13 são apresentadas as ampliações referentes à imagem *Mandrill*.

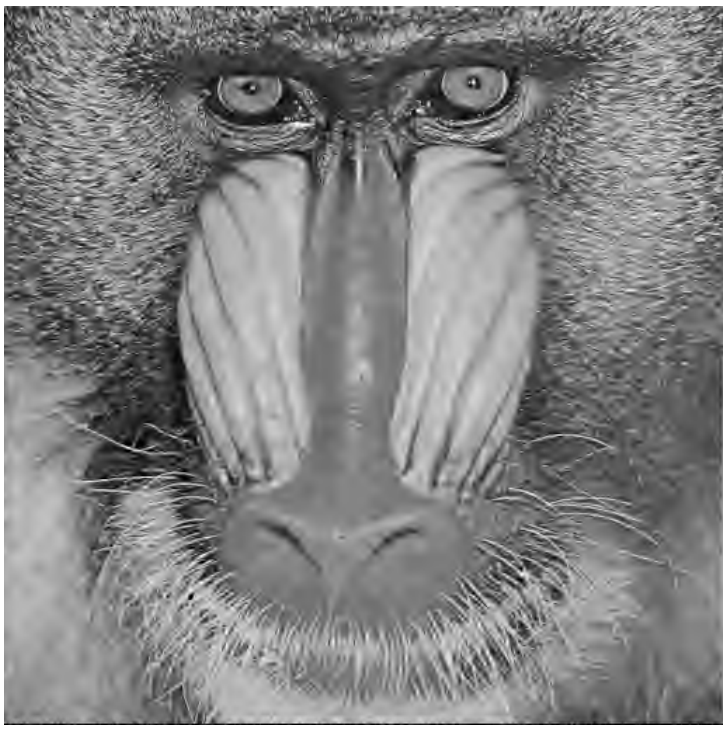
Fig. 13.8 – Ampliações da imagem *Peppers* reconstruída a 0,52bpp:
(Lado esquerdo) ORIGINAL; (Lado direito) TWD+SPIHT.

Fig. 13.9 – Ampliações da imagem *Peppers* reconstruída a 0,52bpp:
(Lado esquerdo) FRACTAL ACELERADO; (Lado direito) FRACTAL-*WAVELET*.

Pode ser observado da Fig. 13.10b que o sistema TWD+SPIHT apresenta bom desempenho visual, em especial na área dos olhos, mas gera muitas distorções de brilho (nos bigodes de *Mandrill*) e perda de textura em áreas como o nariz e as linhas do focinho de *Mandrill*.

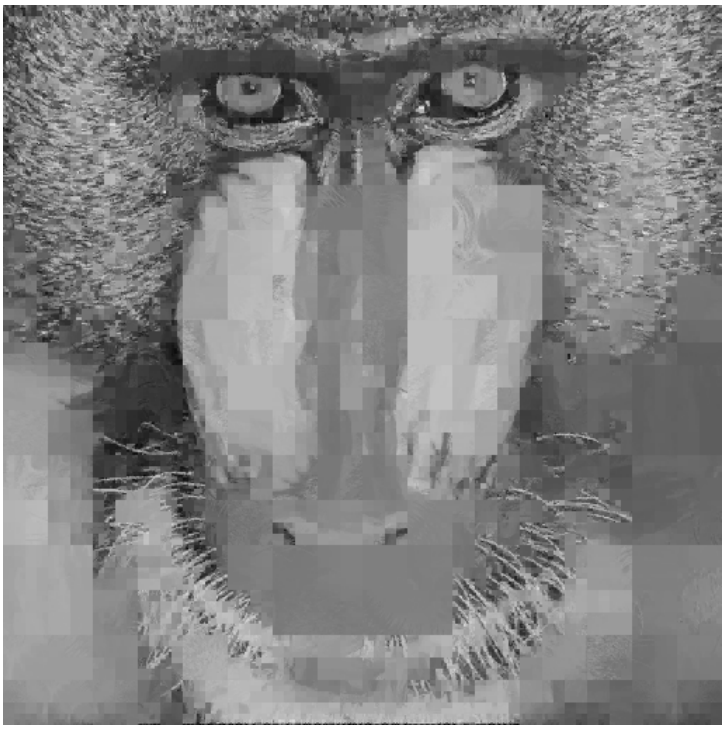


(a)

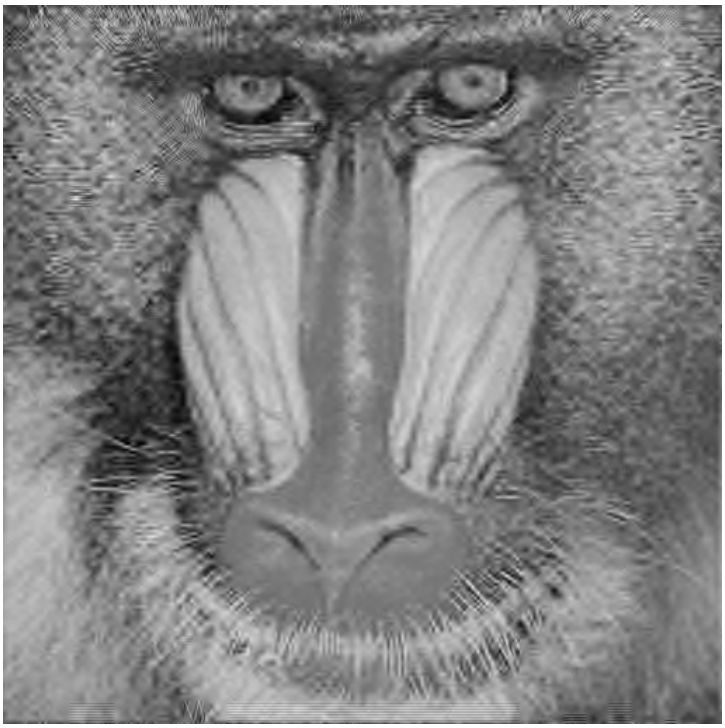


(b) PSNR = 23,75dB

Fig. 13.10 - Imagem *Mandrill* 512x512, 0,52bpp : (a) ORIGINAL; (b) reconstruída TWD+ SPIHT.



(a) PSNR = 21,50 dB



(b) PSNR = 22,38 dB

Fig. 13.11- Imagem *Mandrill* 512x512 reconstruída, 0,52bpp :
(a) FRACTAL ACELERADO; (b) FRACTAL-*WAVELET*.

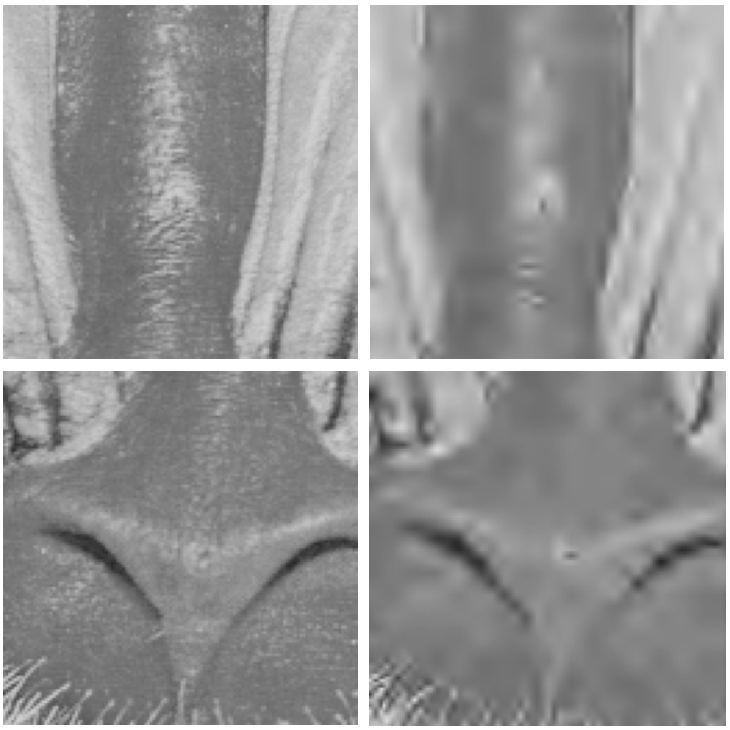


Fig.13.12– Ampliações da imagem *Mandrill* reconstruída a 0,52bpp:
(Lado esquerdo) ORIGINAL; (Lado direito) TWD+SPIHT.

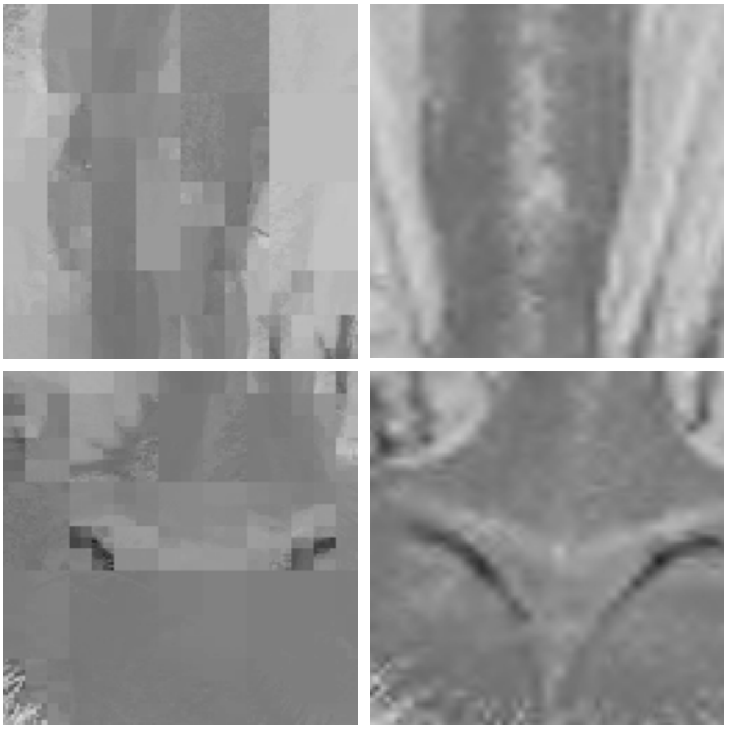


Fig. 13.13 – Ampliações da imagem *Mandrill* reconstruída a 0,52bpp:
(Lado esquerdo) FRACTAL ACELERADO; (Lado direito) FRACTAL-*WAVELET*

O sistema FRACTAL ACELERADO da Fig. 13.11a apresenta novamente excessiva blocagem. O sistema híbrido FRACTAL-*WAVELET* da Fig. 13.11b, apresenta qualidade de imagem superior ao Fractal Acelerado, e similar ao TWD+SPIHT (melhor desempenho em algumas regiões e pior desempenho em outras).

Quanto à velocidade computacional, tem-se que o tempo de processamento para a imagem *Mandrill* foi: 3’32’’ para o TWD+SPIHT; 26’15’’ para o FRACTAL ACELERADO e 5’15’’ para FRACTAL-*WAVELET*. Quanto ao PSNR das imagens reconstruídas, o sistema híbrido FRACTAL-*WAVELET* apresenta PSNR melhor do que o do sistema FRACTAL ACELERADO. O PSNR de TWD+SPIHT é o mais significativo.

As Tabelas XIII.1-XIII.3 apresentam uma gama maior de resultados de tempo de processamento e PSNR para as imagens da Fig. 13.1 a várias taxas de compressão.

TABELA XIII.1 - TWD+SPIHT

Imagem	*Bitrate* (bpp)	PSNR(dB)	Tempo(s)
Lena	0,42	32,79	177
	0,52	32,97	233
	0,73	33,68	365
Peppers	0,44	32,76	198
	0,52	32,65	227
	0,73	33,23	349
Mandrill	0,52	23,75	212
	0,68	24,69	258
	0,85	26,12	470

TABELA XIII.2 - FRACTAL ACELERADO

Imagem	*Bitrate* (bpp)	PSNR(dB)	Tempo(s)
Lena	0,42	31,80	1290
	0,52	32,68	1575
	0,73	34,47	2420
Peppers	0,44	32,87	1502
	0,52	33,86	1806
	0,73	35,46	2829
Mandrill	0,52	21,50	1574
	0,68	22,58	2036
	0,85	24,78	3680

Quanto aos valores de PSNR, pode-se considerar o desempenho médio dos três procedimentos praticamente equivalente em todas as taxas de compressão, com exceção da imagem *Mandrill* (para a qual o PSNR de TWD+SPIHT é melhor). Os artefatos visuais de blocagem, perda

de textura, excesso de brilho e *ringing* apontados na comparação visual apresentada se mantiveram para as demais simulações constantes das Tabelas XIII.1-XIII.3, bem como a característica de velocidade computacional.

Pode-se dizer que, com relação a estes aspectos, o sistema FRACTAL-*WAVELET* apresenta desempenho equivalente ao TWD+SPIHT, evitando, porém, distorções de brilho e textura. Nesse aspecto, o codificador FRACTAL-*WAVELET* reflete realmente sua essência: um intermediário entre técnicas, na medida em que reduz os artefatos de compressão apresentados em cada técnica individualmente.

TABELA XIII.3- FRACTAL -*WAVELET*

Imagem	*Bitrate* (bpp)	PSNR(dB)	Tempo(s)
Lena	0,42	32,00	287
	0,52	33,60	362
	0,73	34,49	641
Peppers	0,44	31,80	304
	0,52	33,80	380
	0,73	35,53	523
Mandrill	0,52	22,38	315
	0,68	23,29	420
	0,85	25,15	738

13.3 – Resultados Estendidos

Os testes da Seção 13.2 feitos com as imagens-teste originais *Lena, Peppers e Mandrill* são estendidos nesta seção, de forma a fornecer uma análise ainda mais completa sobre o desempenho comparativo dos codificadores testados.

Nesta seção (13.3) foram utilizadas todas as 8 imagens 512x512 do conjunto-padrão em escala de cinza disponível no *website* Waterloo Bragzone[1] a várias taxas de compressão.

Considerações e comparações adicionais com outros resultados publicados também foram feitas. Em submissão de um artigo sobre fractais em novembro de 2004, a revista científica *IEEE Transactions on Image Processing* requisitou a comparação com o codificador fractal de Cardinal [4] (referência bastante atualizada) por considerá-lo um bom parâmetro de comparação. Dessa forma, incluímos este codificador como sendo um dos sistemas-base (Sistema-Base 3) a ser comparado com o Codificador Híbrido Proposto por este trabalho.

Os resultados de Cardinal publicados em [4] validaram seu codificador como tendo, na ocasião, desempenho muito bom: proporcionou um aumento significativo de velocidade em suas

comparações para a mesma qualidade de imagem. Esse codificador foi o único sistema-base que *não* foi implementado. As comparações com o Sistema-Base 3 foram feitas baseadas nos dados extraídos diretamente dos gráficos e dados apresentados em [4]. Deve-se frisar que, para essa comparação, utilizou-se rigorosamente o mesmo conjunto de imagens de [4].

As Figs 13.14 e 13.15 apresentam, respectivamente, a curva Taxa-Distorção Média (*average rate-distortion*) e Tempo de Processamento Médio x Taxa de Compressão para o conjunto de imagens testadas. Pode-se notar que as características R-D das Tabelas XIII.1-XIII.3 refletem-se no comportamento das curvas dos gráficos das Figs 13.14 e 13.15, validando as observações da Seção 13.2.

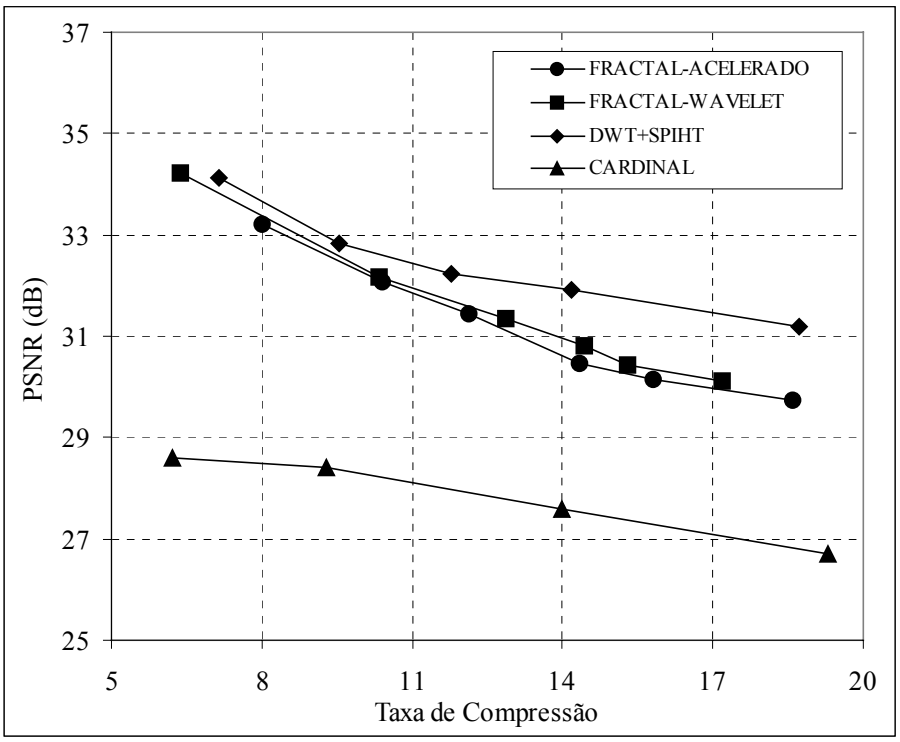


Fig. 13.14 – Curvas de Taxa-Distorção Média para o conjunto *Waterloo Bragzone*.

Deve-se observar que o Codificador Híbrido Fractal-*Wavelet* (bem como os Sistemas-Base de Comparação 1 e 2) superam consideravelmente em termos de PSNR para todos os *bitrates* o Codificador de Cardinal recomendado pela *IEEE*. A explicação mais provável para essa diferença reside na não utilização das transformações geométricas durante o processo de *matching* por Cardinal, limitando a capacidade de procura pelas auto-similaridades.

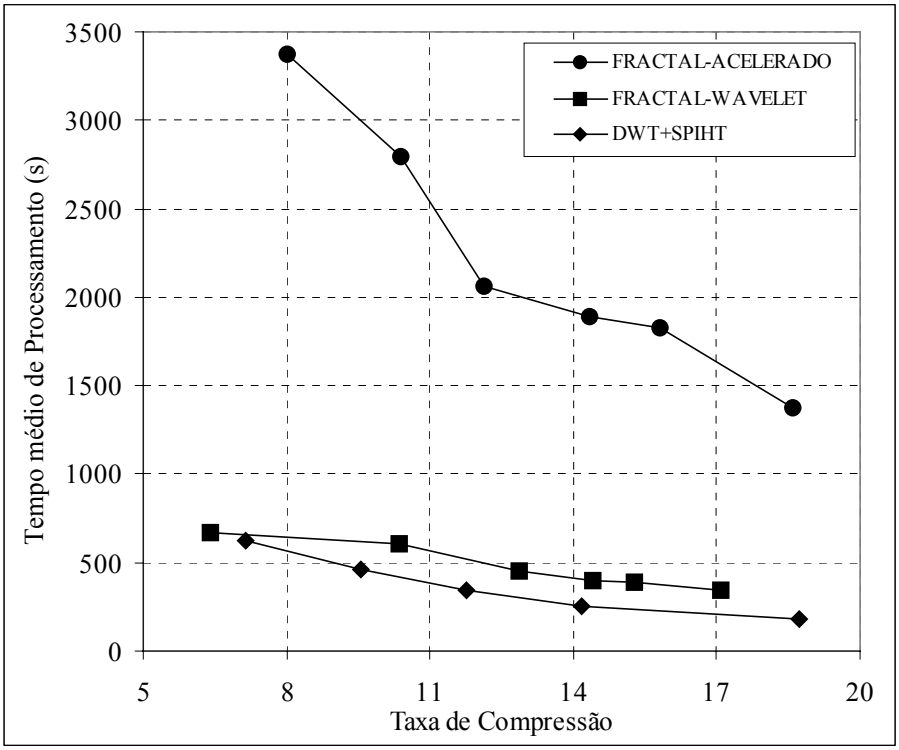


Fig. 13.15 – Curvas de Tempo de Processamento Médio para o conjunto *Waterloo Bragzone*.

Observando as Figs. 13.14 e 13.15, pode-se notar o resultado excelente do ponto de vista de velocidade computacional e PSNR, no qual a velocidade do sistema híbrido proposto (FRACTAL-*WAVELET*) é sempre superior ao FRACTAL ACELERADO, proporcionando uma redução média de 80% no tempo de processamento no procedimento fractal puro para mesmo número de bits, melhor qualidade visual e de PSNR.

O objetivo de minimizar o gasto computacional fractal sem detrimento do PSNR é, portanto, plenamente alcançado através da aplicação da direcionalidade da TWD, tornando viável a exploração da poderosa ferramenta de compressão fractal conjunta.

A importância de se comparar o codificador FRACTAL-ACELERADO e o codificador FRACTAL-*WAVELET* proposto com o TWD+SPIHT se revela no fato deste último ser uma técnica já consolidada e otimizada, enquanto muito pouco do potencial da compressão fractal foi até aqui explorado. Pode-se observar nas Figs 13.14 e 13.15 o quão significativamente o uso da direcionalidade "desloca" positivamente o desempenho da técnica fractal pura (FRACTAL ACELERADO) em direção à excelente técnica de compressão *wavelet* pura (TWD+SPIHT), especialmente no tocante à velocidade de processamento, objetivo central desta pesquisa.

Nota-se que FRACTAL-*WAVELET* apresenta uma redução média de 80% no tempo de processamento de FRACTAL ACELERADO, enquanto o TWD+SPIHT apresenta redução de 86%.

O desempenho visual intermediário do codificador proposto entre o TWD+SPIHT e o FRACTAL ACELERADO (minimizando os artefatos de compressão apresentados individualmente) visto na Seção 13.2 encontra-se refletido no gráfico das Fig.13.14.

A queda mais acentuada da curva de PSNR com o aumento da taxa de compressão ocorre em processamentos que utilizem tecnologia fractal, uma vez que a aplicação desta técnica em compressão de imagens é mais recente e bem menos explorada, necessitando ainda de pesquisas especialmente para otimização dos pontos de operação a altas taxas de compressão.

Uma vez que os resultados de CARDINAL foram tirados diretamente de [4], ou seja, este algoritmo não foi implementado, a curva referente a esse sistema não consta da Fig. 13.15. Para fornecer uma breve análise de tempo de processamento comparativo desse sistema foi levantado o número médio de operações de ponto flutuante por imagem baseado no funcionamento do algoritmo.

Segundo a análise do algoritmo, CARDINAL apresenta, em média, para imagens 512x512 aproximadamente 1,26 GFLOPS (*Giga floating-point operations*) enquanto FRACTAL-ACELERADO usa em média 2,12 GFLOPS para imagens com as mesmas dimensões. Sabe-se que número de GFLOPS por imagem é diretamente proporcional ao tempo de processamento. Assim, como FRACTAL-*WAVELET* reduz o tempo de processamento de FRACTAL ACELERADO em aproximadamente 80%, pode-se estimar que FRACTAL-*WAVELET* reduza o tempo de processamento de CARDINAL em cerca de 66 %.

13.4 – Comentários

Neste capítulo foram apresentados os resultados experimentais do codificador híbrido FRACTAL-*WAVELET*, cujo objetivo era minimizar o gasto computacional fractal na compressão de imagens estáticas através da aplicação da TWD e suas propriedades, sem que houvesse detrimento do PSNR ou da qualidade visual.

No total dos experimentos foram utilizadas 8 imagens 512x512 do conjunto-padrão em escala de cinza disponível no *website* Waterloo Bragzone[1]. Para os testes e simulações foi utilizado um computador Pentium 4, 2.6GHz com 1Gbyte de RAM. Os resultados experimentais do codificador híbrido apresentados foram comparados em termos de tempo de processamento, qualidade visual

das imagens recuperadas, PSNR e número de bits com os seguintes sistemas:

1) Sistema-Base 1: Codificador *Wavelet* Puro (“TWD+SPIHT”);
2) Sistema-Base 2: Codificador de Fisher (“FRACTAL ACELERADO”) [3];
3) Sistema-Base 3: Codificador de Cardinal (“CARDINAL”) [4] .

Na questão da análise visual, o codificador FRACTAL-*WAVELET* refletiu realmente sua essência: um intermediário entre técnicas, na medida em que reduziu os artefatos de compressão apresentados em cada técnica individualmente (efeito de bloco fractal e distorções de brilho e textura *wavelet*). O desempenho visual de FRACTAL-*WAVELET* foi similar ao desempenho de TWD+SPIHT e ambos consideravelmente superiores à FRACTAL ACELERADO.

Segundo os resultados obtidos, pôde-se observar que FRACTAL-*WAVELET* proporcionou uma excelente redução média de 80% no tempo de processamento no procedimento fractal puro para mesmo número de bits e melhor qualidade visual e de PSNR.

O uso da direcionalidade “deslocou” positivamente o desempenho de FRACTAL ACELERADO em direção à excelente técnica já consolidada TWD+SPIHT, especialmente no tocante à velocidade de processamento.

FRACTAL-*WAVELET* também reduz o tempo de processamento de CARDINAL (referência *IEEE*) em cerca de 66 % para PSNR consideravelmente superior.

O objetivo de minimizar o gasto computacional fractal sem detrimento do PSNR foi, portanto, plenamente alcançado através da aplicação da direcionalidade da TWD, tornando viável a exploração da ferramenta conjunta.

Nenhum artefato de blocagem (comum a procedimentos fractais puros) resultou do processo de compressão híbrido. Os detalhes da imagem e as características de transmissão progressiva *wavelet* foram também preservados. O aumento na complexidade do codificador pode ser considerado de pequeno porte.

Capítulo 14

Conclusões e Sugestões para Trabalhos Futuros

14.1 – Comentários sobre o Contexto e Contribuições

Embora os métodos de compressão sejam hoje de uso extensivo, o ritmo acelerado no crescimento das aplicações de imagens e vídeos digitais impõem ao mundo tecnológico sua necessidade pelo aumento na capacidade de armazenamento e na velocidade de transmissão.

Novas tecnologias, tais como fractais e *wavelets* surgem como aliadas potenciais no estabelecimento de novos padrões de compressão de imagens, dadas as suas vantajosas características.

A codificação fractal de imagens consiste, na prática, em representar os blocos da imagem através de coeficientes de transformações contrativas, explorando-se o conceito de auto-similaridade dentro da imagem. Assim, nesse tipo de codificação, ao invés de armazenar/transmitir os blocos da imagem como uma coleção de *pixels*, somente são enviados/armazenados os coeficientes destas transformações. Este princípio de funcionamento permite obter altas taxas de compressão mantendo-se ótima qualidade visual.

Entretanto, para determinar as auto-similaridades, cada *range-block NxN* é comparado a todos os *domain-blocks 2Nx2N* da *Domain Pool* dentro de uma imagem *MxM*. Como conseqüência, os algoritmos de *matching* fractais têm uma complexidade computacional da ordem de $O[M^4]$, o que demanda um tempo de processamento exaustivo, tipicamente levando horas.

Diversas tentativas recentes têm sido feitas na tentativa de minimizar o extenso tempo de processamento fractal, abordando desde vetores de características até redes neurais. Basicamente consistem em modificar os seguintes aspectos: a composição da *domain pool*, o tipo de procura usado no *block matching*, ou a representação/quantização dos parâmetros transformados. Trabalhos recentes podem ser encontrados em [4][62]-[71].

Entretanto, muita pesquisa ainda se faz necessária para destravar o potencial fractal latente para a compressão de imagens. Sem dúvida nenhuma, essas pesquisas devem residir na questão de promover a aceleração da etapa de compressão dessa nova tecnologia.

Nesse contexto, mesclando as duas poderosas técnicas *wavelet* e fractal, o objetivo deste trabalho é propor um método de aceleração para a exaustiva compressão fractal de imagens

estáticas, através da aplicação da TWD e suas propriedades sem que haja detrimento do PSNR ou da qualidade visual. Logo, este trabalho apresenta um novo Codificador Híbrido Fractal-*Wavelet*, que aplica a compressão fractal acelerada à imagens estáticas decompostas pela transformada *wavelet*, explorando a direcionalidade das subimagens-*wavelet*, constatada por Shapiro [2].

Dessa forma, são sumarizadas aqui as principais contribuições deste trabalho como sendo:

(a) Idéia Central: Exploração conjunta da codificação fractal com a direcionalidade *wavelet*. Conforme foi abordado no Capítulo 12, Shapiro coloca que as subbandas-*wavelet* de um mesmo tipo representam uma mesma estrutura espacial da imagem em diferentes escalas [2]. Logo, neste trabalho, propôs-se a idéia de que os pares *range-domain* fractais de um determinado tipo de subimagem *wavelet* poderiam ser preditos pelos pares *range-domain* da subimagem-*wavelet* de menor escala do mesmo tipo. Essa direcionalidade substitui o uso de *Domain Pools* de forma bastante significativa e eficiente. A conseqüência direta foi o aumento de velocidade de compressão.

(b) Implementação dos Sistemas-Base Fractal Puro e *Wavelet* Puro (para comparação). Elaboração e implementação do Codificador/Decodificador Híbrido Fractal-*Wavelet* completo, contando com um esquema de controle de qualidade (que classifica os blocos para uma melhor distribuição de quantização, permitindo que haja uma melhor codificação dos blocos conforme a necessidade) e com um mecanismo de escape (prevenindo exceções de ausência de auto-similaridade).

Através desse esquema, para diversas imagens e *bitrates*, obteve-se uma melhora visual corroborada pelos valores de PSNR e um aumento na velocidade de processamento fractal pura em cerca de 80% para uma mesma taxa de bits, tornando viável a exploração dessa poderosa ferramenta de compressão fractal conjunta. Nenhum artefato de blocagem (comum a procedimentos fractais puros) resultou do processo de compressão híbrido. Os detalhes da imagem e as características de transmissão progressiva *wavelet* foram também preservados. O aumento na complexidade do codificador pode ser considerado de pequeno porte.

14.2 – Conclusões Breves sobre os Capítulos e Relação com a Proposta

Este trabalho apresentou três macrodivisões, a primeira correspondendo ao estudo da teoria dea transformada *wavelet* (Capítulos 2 a 7); a segunda correspondendo ao estudo da tecnologia

fractal (Capítulos 8 a 11) e a terceira correspondendo à apresentação do codificador proposto, seus resultados e avaliações pertinentes (Capítulos 12 a 14).

Primeiramente, o Capítulo 2 apresentou os princípios básicos da transformada *wavelet* contínua 1D e 2D, suas características e princípios de funcionamento. No Capítulo 3 foi realizado um breve estudo acerca da codificação por subbanda resultando na implementação prática da tranformada *wavelet* discreta (TWD) através dos filtros do algoritmo rápido de Mallat. Trata-se de uma preparação para o Capítulo 4, no qual foram apresentados os sinais resultantes da decomposição *wavelet*: o sinal passa-baixas (PB) e os sinais de detalhes. Sobre esses sinais (estendidos para o caso 2D e biortogonal) foi aplicada a codificação fractal do codificador híbrido proposto nesta pesquisa.

Dessa forma, em resumo, o Capítulo 4 apresentou a decomposição e síntese da TWD para o caso ortogonal, bem como os sinais PB e de detalhes. Uma vez colocada a teoria ortogonal, pôde-se relaxar as condições de operação, apresentando a biortogonalidade, que sana as deficiências do caso ortogonal e possibilita efetivamente a aplicação da TWD na compressão de imagens com excelente desempenho.

Esse desempenho se deve às excelentes características conjuntas dos filtros biortogonais: suporte compacto, simetria e bom compromisso entre regularidade e comprimento do filtro. A biortogonalidade consagrou o uso da TWD na compressão de imagens e é o tema do Capítulo 5.

Foi apresentado a seguir, no Capítulo 6, a análise multiresolução bidimensional. Foram então detalhadas a decomposição e síntese da TWD para o caso bidimensional ortogonal, bem como os sinais PB (2D) e de detalhes (2D) resultantes do processo de decomposição.

Encerrando a teoria de *wavelets*, o Capítulo 7 apresentou as características desejáveis para a TWD na compressão de imagens (em especial a regularidade) culminando com a importância da biortogonalidade e com as características da *spline* biortogonal 9-7, que foi adotada por este trabalho. Também foi apresentada a codificação SPIHT (*Set Partitioning in Hierarchical Trees*), que compôs o codificador *wavelet* puro utilizado como o Sistema-Base 1 que foi comparado com o codificador híbrido proposto nesta pesquisa.

Este capítulo finaliza a abordagem da teoria de transformada *wavelet*, na qual foram fundamentados todos os conceitos necessários à apresentação da parcela *wavelet* do codificador híbrido proposto neste trabalho.

O Capítulo 8 abordou de forma ilustrativa as bases da compressão fractal de imagens. Foram abordados os conceitos de transformação afim, iteração e atrator. Esse capítulo introduziu

uma abordagem inicial para que, num segundo momento (Capítulos 9 e 10), fosse colocada a abordagem matemática responsável por esses fundamentos.

Dessa forma, no Capítulo 9 apresentou-se em destaque os teoremas da colagem e do mapeamento contrativo referentes à teoria de espaços métricos completos. O primeiro deles permite que, dado o atrator, possa-se determinar os coeficientes das transformações afins a serem enviados (codificação fractal). O segundo garante que o processo fractal converge para um atrator (decodificação fractal). Viu-se ainda nesse capítulo o método de geração de fractais conhecido como IFS (*Iterated Function System*), inspiração dos codificadores PIFS.

O Capítulo 10 encerrou a abordagem matemática comentando o PIFS (*Partitioned Iterated Function Systems*), método efetivamente usado nos codificadores fractal atuais, incluindo o codificador híbrido desta pesquisa. Foram apresentados os princípios de funcionamento (baseado na similaridade por partes), o conceito de *Domain Pool*, o particionamento da imagem em *domain-blocks* e *range-blocks*, a aplicação das transformações afins, o método básico de busca e encontro do *domain-block*-equivalente e as informações a serem enviadas ao decodificador.

Encerrando a teoria fractal, o Capítulo 11 teve por tema a codificação fractal acelerada, abordando os trabalhos atuais da literatura científica e abordando os dois Sistemas-Base Fractais para comparação com o codificador híbrido proposto. Ambos os Sistemas-Base (o Codificador de Fisher [3] e o Codificador de Cardinal [4]) também objetivam acelerar a velocidade de codificação fractal.

O primeiro deles foi escolhido por ser um dos codificadores mais implementados atualmente. O segundo foi escolhido por ser citado como um bom parâmetro de comparação pela revista científica *IEEE Transactions on Image Processing* em dezembro de 2004.

Esse capítulo encerrou a abordagem da teoria fractal, na qual foram fundamentados os conceitos necessários à apresentação da parcela fractal do codificador híbrido proposto.

No Capítulo 12 foram detalhados os princípios de funcionamento do Codificador Hìbrido *Fractal-Wavelet* Proposto, cuja idéia central foi a exploração conjunta da codificação fractal com a direcionalidade *wavelet*, procedimento que substitui o uso de *Domain Pools* de forma eficiente. A conseqüência direta é o aumento de velocidade de codificação. As particularidades do Codificador Proposto e dos Sistemas-Base de Comparação também foram alvo deste capítulo.

Finalmente, o Capítulo 13 apresentou os resultados e análises das simulações realizadas. Os resultados do Codificador Híbrido foram comparados com os resultados de Sistemas-Base Fractais

"puros" e com o Sistema-Base *Wavelet* Puro em termos de tempo de codificação, PSNR, e qualidade subjetiva para uma ampla gama de imagens codificadas a vários *bitrates*.

Finaliza-se este trabalho apresentando, nesse capítulo, os comentários e conclusões à luz dos resultados obtidos, analisando-se as perspectivas de melhorias e adaptações do procedimento proposto.

14.3 – Comentários sobre Simulações e Resultados

Os codificadores testados (Sistemas-Base e Codificador Proposto) foram implementados em Matlab 6.1, que também foi o ambiente de programação para a visualização de imagens e construção de gráficos. Os tempos de processamento englobaram os valores de codificação e decodificação.

Para os resultados apresentados no Capítulo 13, foram utilizadas 8 imagens 512x512 do conjunto-padrão em escala de cinza disponível no *website* Waterloo Bragzone[1]. Essas imagens são extensamente utilizadas na literatura científica de processamento digital de imagens.

Embora sistemas de compressão de imagens completos empreguem codificação entrópica e pós-processamento nas imagens decodificadas para melhorar os resultados, nesta pesquisa esses procedimentos não foram adotados, uma vez que o objetivo era focar especificamente nas implementações fractais e *wavelet*. Essa opção permitiu o isolamento de variáveis e uma melhor análise do núcleo dos processos. Entretanto, para a estimação do *bitrate* das imagens codificadas, foi utilizado o cálculo de entropia de primeira ordem.

Os resultados experimentais do codificador híbrido ("FRACTAL-*WAVELET*") apresentados foram comparados em termos de tempo de processamento, qualidade visual das imagens recuperadas, PSNR e número de bits com os seguintes sistemas:

1) Sistema-Base 1: Codificador *Wavelet* Puro ("TWD+SPIHT");
2) Sistema-Base 2: Codificador de Fisher ("FRACTAL ACELERADO") [3];
3) Sistema-Base 3: Codificador de Cardinal ("CARDINAL") [4] .

Quanto à análise visual das imagens, o codificador híbrido proposto (FRACTAL-*WAVELET*) refletiu realmente sua essência: um intermediário entre a técnica fractal e entre a técnica *wavelet*, na medida em que reduziu os artefatos de compressão apresentados em cada técnica individualmente (efeito de bloco fractal e distorções de brilho e textura *wavelet*). O desempenho visual de FRACTAL-*WAVELET* foi similar ao desempenho de TWD+SPIHT e ambos

consideravelmente superiores à FRACTAL ACELERADO.

Segundo os resultados obtidos, pôde-se observar que FRACTAL-*WAVELET* proporcionou uma excelente redução média de 80% no tempo de processamento no procedimento fractal puro para mesmo número de bits e melhor qualidade visual e de PSNR.

O uso da direcionalidade "deslocou" positivamente o desempenho de FRACTAL ACELERADO em direção à excelente técnica já consolidada TWD+SPIHT, especialmente no tocante à velocidade de processamento.

FRACTAL-*WAVELET* também reduziu o tempo de processamento de CARDINAL (referência *IEEE* de 2004) em cerca de 66 % para PSNR consideravelmente superior.

Pôde-se concluir que o objetivo de minimizar o gasto computacional fractal sem detrimento do PSNR foi, portanto, plenamente alcançado através da aplicação da direcionalidade da TWD, tornando viável a exploração da ferramenta híbrida.

Nenhum artefato de blocagem (comum a procedimentos fractais puros) resultou do processo de compressão híbrido. Os detalhes da imagem e as características de transmissão progressiva *wavelet* foram também preservados no codificador proposto. O aumento na complexidade do codificador pôde ser considerado de pequeno porte, tratando-se simplesmente da inserção da decomposição e recosntrução *wavelet*.

14.4 – Sugestões para Trabalhos Futuros

Algumas sugestões que foram visualizadas durante a realização desta pesquisa foram:

- Testar o codificador híbrido proposto aplicando-se outros tipos de *wavelets* (que não a *spline* 7-9) com propriedades diferentes, comparando-se sua influência na velocidade computacional e no desempenho geral.
- Aplicação da técnica de *lifting* para aceleração da decomposição e reconstrução da parcela *wavelet* do codificador proposto;
- Usar outros métodos de classificação de domínios diferente do método de Fisher e observar sua influência nas curvas de taxa-distorção e de velocidade.
- Implementação de técnicas de codificação entrópica.
- Num nível mais complexo, a extensão para imagens coloridas e adequação para sinais de vídeo. Deve-se considerar as implicações das taxas temporais típicas de sinais de vídeo se

comparada às taxas de tempo de processamento fractais.

Assim, foram listadas as principais contribuições e resultados do presente trabalho e foram feitas algumas sugestões que espera-se, possam auxiliar o desenvolvimento de outras pesquisas relacionadas com o processamento e a compressão digital de imagens e vídeo.

Conforme dito anteriormente, o exaustivo tempo de processamento das implementações fractais correntes restringem essa técnica à aplicações de arquivamento, tais como enciclopédias digitais. Métodos *wavelet* são, comparativamente, mais adequados para aplicações que requerem codificação rápida, tais como comunicações em tempo real via *Internet*. Métodos híbridos rápidos como o codificador proposto nesta pesquisa, podem permitir a extensão do uso das técnicas fractais aos meios rápidos de comunicação.

Referências Bibliográficas

- *CSVT : Circuits and Systems for Video Technology*
- *PAMI : Pattern Analysis and Machine Intelligence*

[1] S. T. Wealsted – "Fractal and Wavelet Image Compression Techniques" – Ed. SPIE Optical Engineering Press, Washington, USA, 1999.

[2] J. M. Shapiro – "Embedded Image Coding Using Zerotrees of Wavelet Coefficients" – *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3445-3462, Dez., 1993.

[3] Y. Fisher – "Fractal Image Compression, Theory and Application" – Ed. Springer-Verlag, New York Inc, USA, 1995.

[4] J. Cardinal – "Fast Fractal Compression of Greyscale Images" – *IEEE Transactions on Image Processing*, vol. 10, no. 01, pp. 159-164, Jan., 2001.

[5] K. R. Castleman – "Digital Image Processing" – Ed. Prentice Hall, New Jersey, USA, 1996.

[6] M. C. Q. Farias – "Aplicação da Transformada Wavelet na Compressão de Imagens" – Dissertação de Mestrado, FEEC/UNICAMP, SP, Brasil, Jun., 1998.

[7] A. Grossman; J. Morlet – "Decomposition of Hardy Functions into Square Integrable Wavelets of Constant Shape" – *SIAM J. Applied MAthematics*, vol. 15, pp. 726-736, 1984.

[8] C. F. Chui – "An Introduction to Wavelets" – Academic Press Inc., San Diego, USA, 1992.

[9] I. Daubechies – "Ten Lectures on Wavelets" – SIAM, Philadelphia, USA, 1992.

[10] S. Mallat – "A Theory for Multiresolution Signal Decomposition: the Wavelet Representation" –*IEEE Transactions on PAMI*, vol. 11, pp. 674-693, 1989.

[11] J. W Woods; S. D. O'Neill – "Subband Coding of Images" –*IEEE Transactions on ASSP*, vol. 34, pp. 1278-1288, 1986.

[12] S. Mallat – "Multifrequency Channel Decompositions of Images and Wavelet Models" –*IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, no.12, Dez., 1989.

[13] E. H. Adelson; E. Simoncelli; R. Hingorani – "Orthogonal Pyramid Transforms for Image Coding" – *Proceedings of SPIE*, vol. 845, pp. 50-58, Out., 1987.

[14] B. Porat – "Digital Processing of Random Signals" – Ed. Prentice Hall, Pennsylvania, USA, 1992.

[15] M. Antonini; M. Barlaud; P. Mathiew; I. Daubechies – "Image Coding Using Wavelet Transform" – *IEEE Transactions on Image Processing*, vol. 01, no. 02, Abr., 1992.

[16] I. Daubechies – "Orthogonal Bases of Compactly Supported Wavelets" – *Communications on Pure Applied Mathematics*, vol. XLI, pp. 990-996, 1988.

[17] A. Cohen; J. C. Feauveau; I. Daubechies – "Biorthogonal Bases of Compactly Supported Wavelets" – *Communications on Pure Applied Mathematics*, vol. 45, pp. 485-560, 1992.

[18] L. Prasad; S. S. Iyengar – "Wavelet Analysis With Applications to Image Processing" – Ed. CRC Press, Florida, USA, 1997.

[19] S. Mallat – "A Wavelet Tour of Signal Processing" – Ed. Academic Press, San Diego, USA, 2001.

[20] J. N. Bradley; C. M. Brinslaw; T. Hopper – "The FBI Wavelet/Scalar Quantization Standard for Grayscale Digitized Fingerprints Image Compression" – *Visual Info. Process II – SPIE* Florida, USA, Abr, 1993.

[21] S. Mallat – "Multiresolution Approximation and Wavelets" –*IEEE Transactions on PAMI*, vol. 11, no.07, Jul, 1989.

[22] C. Chrysafis; A. Ortega – "Line Based, Reduced Memory, Wavelet Image Compression" –*IEEE Transactions on Image Processing*, vol. 9, no.03, pp. 378-389, Mar, 2000.

[23] C. Chrysafis; A. Ortega – "Efficient Context-Based Entropy-Coding for Lossy Wavelet Image Compression" – *Proc. IEEE Data Compression Conf.*, Snowbird, UT, pp. 241-250, 1997.

[24] R. W. Bucigrossi; E. P. Simoncelli – "Image Compression via Joint Statistical Characterization in the Wavelet Domain" – *IEEE Transactions on Image Processing*, vol. 8, no.12, pp. 1688-1701, Dez, 1999.

[25] M. Ohta; S. Nakagi – Hybrid Picture Coding with Wavelet Transform and Overlapped Motion-Compensated Interframe Prediction Coding" – *IEEE Transactions on Signal Processing*, vol.41, no.12, pp.3416-3424, Dez. 1993.

[26] J. Vass; B. –B. Chai; K. Palaniappan; X. Zhuang – "Significance-Linked Connected Component Analysis for Very Low Bit-Rate Wavelet Video Coding" – *IEEE Transactions on CSVT*, vol.9 no.4, pp. 630-647, Jun 1999.

[27] X. Yang; K. Ramchandran – "Scalable Wavelet Video Coding Using Aliasing-Reduced Hierarchical Motion Compensation" – *IEEE Transactions on Image Processing*, vol.9 no.5, pp. 778-791, Maio 2000.

[28] Y. Q. Zhang; S. Zafar – "Motion-Compensated Wavelet Transform Coding for Color Video Compression" – *IEEE Transactions on CSVT*, vol.2, no.3, pp.285-296, Set 1992.

[29] F. A. Mujica; J. –P. Leduc; R. Murenzi; M.J.T. Smith – "A New Motion Parameters Estimation Algorithm Based on the Continuous Wavelet Transform" – *IEEE Transactions on Image Processing*, vol.9 no.5, pp. 873-888, Maio 2000.

[30] H. -W. Park; H.-S. Kim – " Motion Estimation Using Low-Band-Shift Method for Wavelet-Based Moving Picture Coding" – *IEEE Transactions on Image Processing*, vol.9 no.4, pp. 577-587, Abr 2000.

[31] I. E. G. Richardson – "H.264 and MPEG-4 Video Compression" – Ed. John Wiley & Sons, Chichester, UK, 2003.

[32] M. Vetterli; C. Herley – " Wavelets and Filter Banks: Relationships and New Result" – *Proc. IEEE ICASSP*, Alburquerque, USA, Abr 2000.

[33] V. I. B. Sablón; Y. Iano – "Processamento e Compressão do Sinal de Vídeo Utilizando a Transformada Wavelet" – Dissertação de Doutorado, FEEC/UNICAMP, SP, Brasil, Dez., 2002.

[34] E. H. Adelson; P. Burt – "The Laplacian Pyramid as a Compact Image Code" – *IEEE Transactions Commun*, vol. 31, pp. 482-540, Out., 1983.

[35] I. Daubechies – "Orthogonal Bases of Compactly Supported Wavelets II – Variations on a Theme" – *ATT&Bell Lab.*, *Tech. Report* TM 11217-89111617, USA, 1990.

[36] A. Said; W. A. Pearlman – "Image Compression Using The Spatial-Orientation Tree" – *IEEE Int. Symp. Circ. and Syst*, Chicago, IL, USA, pp. 279-282, Maio, 1993.

[37] A. Said; W. A. Pearlman – "A New, Fast and Efficient Image Coded Based on Set Partitioning in Hierarchical Trees" –. *IEEE Transactions on CSVT*, vol. 6, no. 3, pp. 243-250, Jun, 1996.

[38] P. N. Topiwala – "Wavelet Image and Video Compression" – Ed. Kluwer Academic Publishers, Massachusetts, USA, 1998.

[39] J. M. Shapiro – "Embedded Image Coding Using Zerotrees of Wavelet Coefficients" – *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3445-3462, Dez, 1993.

[40] "Handbook of Image and Video processing" – Ed. Al Bovik, Academic Press, San Diego, USA, 2000.

[41] K. R. Rao; P. C. Yip – "The Transform and Data Compression Handbook" – Ed. CRC Press, LLC, USA, 2001.

[42] K. Sayood – "Introduction to Data Compression" – 2nd ed, Ed. Morgan Kaufmann, CA, USA, 2000.

[43] J. Tian; R. O. Wells Jr – "A Lossy Image Codec Based on Index Coding" – *IEEE Data Compression Conference*, DCC'96, pp. 456, 1996.

[44] J. Tian; 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.

[45] J. Tian; 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.

[46] J. S. Walker – "A Lossy Image Codec Based on Adaptatively Scanned Wavelet Difference Reduction" – *Optical ENgineering*, in press.

[47] B. Mandelbrot – "The Fractal Geometry of Nature" – Ed. W. H. Freeman and Company, New York, USA, 1983.

[48] M. F. Barnsley; A. Sloan – "A better Way to Compress Images" – *Byte*, pp. 215-223, Jan, 1988.

[49] M. F. Barnsley; A. Sloan – "Method and Apparatus for Image Compression by Iterated Funcion System" – *US Patent # 4.941.193*, 1990.

[50] M. F. Barnsley; A. Sloan – "Method and Apparatus for Processing Digital Data" – *US Patent # 5.065.447*, 1991.

[51] A. Jacquin – "Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations" – *IEEE Transactions on Image Processing*, vol. 01, no. 01, pp. 18-30, 1992.

[52] N. Lu – "Fractal Imaging" – Academic Press, San Diego, USA, 1997;

[53] J. E. Hutchinson – "Fractal and Self-Similarity" – *Indiana University Mathematics Journal*, Vol. 35, No. 5, 1981.

[54] M. F. Barnsley; A. Jacquin – "Applications of Recurrent Iterated Function Systems to Images" – *SPIE Visual Communications and Image Processing* , pp. 122-131, 1998.

[55] E. L. Lima – "Espaços métricos" - Ed. Projeto Euclides – Rio de Janeiro, Brasil, 1993.

[56] M. F. Barnsley– "Fractals Everywhere 2nd edition" – Ed. Morgan Kaufmann, Academic Press, San Diego, USA, 1993.

[57] M. J. Turner; J. M. Blackledge; P. R. Andrews– "Fractal Geometry in Digital Imaging" – Academic Press, San Diego, USA, 1993.

[58] P. R. Massopust– "Fractal Functions, Fractal Surfaces, and Wavelets" – Academic Press, San Diego, USA, 1994.

[59] M. Peruggia– "Discrete Iterated Function Systems" – Ed. A K Peters, Massachusetts, USA, 1993.

[60] W. Rudin – "Real and Complex Analysis" – Ed. Maximillian, New York, USA, 1972.

[61] A. Jacquin – "A Fractal Theory of Iterated Markov Operators with Applications to Digital Image Compression" – PhD. Thesis, Georgia Institute of Technology, USA, 1989.

[62] C. S. Tong; M. Wong – "Adaptative Approximate Nearest Neighbor Search for Fractal Image Compression" – *IEEE Transactions on Image Processing*, vol. 11, No.06, pp. 605-615, Jun, 2002.

[63] C. S. Tong; M. Pi – "Fast Fractal Image Encoding Based on Adaptative Search" – *IEEE Transactions on Image Processing*, vol. 10, No.09, pp. 1269-1277, Set, 2001.

[64] C. -C. Wang; C. –H. Hsieh – "An Efficient Fractal Image Coding Method Using Interblock Correlation Search" – *IEEE Transactions on CSVT*, vol. 11, No.02, pp. 257-261, Fev, 2001.

[65] D. Saupe – "Accelerating Fractal Image Compression by Multi-Dimensional Nearest Neighbor Search" –*Proceedings DCC'95 Data Compression Conference*, pp. 222-231, Mar 1995.

[66] C. K. Lee; W. K. Lee – "Fast Fractal Image Block Coding Based on Local Variances" – *IEEE Transactions on Image Processing*, vol. 7, pp. 888-891, Jun, 1998.

[67] M. Polvere; M. Nappi – "Speed-Up in Fractal Image Coding Comparison of Methods" – *IEEE Transactions on Image Processing*, vol. 09, pp. 1002-1009, Jun, 2000.

[68] T. –K. Truong; J. -H, Jeng; I. S. Reed; P. C. Lee; A. Q. Li – "A Fast Encoding Algorithm for Fractal Image Compression Using DCT Inner Product" - *IEEE Transactions on Image Processing*, vol. 9, No.4, pp. 529-534, Abr, 2000.

[69] B. B. Eqbal – "Enhancing the Speed of Fractal Image Compression" – *Opt. Eng*, vol. 34, No.06, pp. 1705-1710, Jun, 1995.

[70] S. Lee; S. Ra – "An Analysis of Isometry Transforms in Frequency Domain for the Fast Fractal Coding" – *IEEE Signal Proc. Lett*, vol. 6, pp. 100-102, Maio, 1999.

[71] E. W. Jacobs, R. D. Ross; Y. Fisher – "Fractal Image Compression using Iterated Transforms" – *Technical Report 1408*, Naval Ocean Systems Center, San Diego, USA, 1991.

[72] E. W. Jacobs, R. D. Ross; Y. Fisher – "Fractal-based Image Compression II" – *Technical Repot 1362*, Naval Ocean Systems Center, San Diego, 1990.

[73] R. F. Sproull – "Refinements to Nearest Neighbor Searching in K-Dimensional Trees" – *Algorithmica*, vol. 6, pp. 579-589, 1991.

[74] A. Bogdan; H. Meadows – "Kohonen Neural Network for Image Coding Based on Iteration Transformation Theory" – *Proceedings of SPIE*, vol. 1766, pp. 425-436, 1992.

[75] D. McGregor; R. J. Fryer; W. P. Cookshott; P. Murray – "Fast Fractal Transform Method for Data Compression" – *University of Strachclyde Research Report*, 94/156 [IKBS-17-94], 1994.

[76] J. Kominek – "Algorithm for Fast Fractal Image Compression" – *Proceedings of SPIE Symp. Electronic Imaging: Science Technology*, vol. 2419, 1995.

[77] B. E. Wohlberg; G. de Jager – "Fast Image Domain Fractal Compression by DCT Domain Block Matching" – *Electron. Lett.*, vol. 31, pp. 869-870,1995.

[78] I. Katsavounidis; C.-C. J. Kuo; Z. Zhang – "Fast Tree-Structured Nearest Neighbor Encoding for Vector Quantization" – *IEEE Transactions on Image Processing*, vol. 5, pp. 398-404, Fev, 1996.

[79] C. Caso; C. -C. J. Kuo – "New Results for Fractal/Wavelet Image Compression" – *Proceedings of SPIE Visual Communications and Image Processing*, vol. 2727, pp. 536-547, Mar, 1996.

[80] R. Rinaldo; G. Calvagno – "Image Coding by Block Prediction of Multiresolution Subimages" – *IEEE Transactions on Image Processing*, vol. 4, no. 07, pp. 909-920, Jul, 1995.

[81] G. M. Davis – "A Wavelet Based Analysis of Fractal Image Compression" – *IEEE Transactions on Image Processing*, vol. 7, no. 02, pp. 141-154, Fev, 1998.

[82] S. Asgari; T. Q. Nguyen; W. A. Sethares – "Wavelet-Based Fractal Transforms for Image Coding With no Search" – *Proc IEEE International Conf. On Image Processing*, ICIP97, 1997.

[83] D. Herbert; E. Soundarajan – "Fast Fractal Image compression With Triangulation Wavelets " – *Proc. SPIE Conf. On Wavelets Applications in Signal and Image Processing VI*, San Diego, USA, 1998.

[84] G. M. Davis – "Adaptative Self-Quantization of Wavelet Subtrees: A Wavelet-Based Theory of Fractal Image Compression " – *Proc. SPIE Conf. On Wavelets Applications in Signal and Image Processing III*, San Diego, USA, 1995.

[85] G. M. Davis – "Implicit Image Models for Fractal Image Compression " – *Proc. SPIE Conf. On Wavelets Applications in Signal and Image Processing IV*, Denver, USA, 1996.

[86] J. Li; C. –C. J. Kuo – "Image Compression With a Hybrid Wavelet-Fractal Coder" – *IEEE Transactions on Image Processing*, vol. 8, no. 06, pp. 868-874, Jun, 1999.

[87] Y. –Q. Zhang; S. Zafar – "Motion-Compensated Wavelet Transform Coding for Color Video Compression" – *IEEE Transactions on CSVT*, vol. 2, no. 03, pp. 285-296, Set, 1992.

[88] J. Zan; O. Ahmad; M. N. S. Swamy – "New Techniques for Multiresolution Motion Estimation" – *IEEE Transactions on CSVT*, vol. 12, no. 09, pp. 793-802, Set, 2002.

[89] S. Kim; S. Rhee; J. G. Jeon; K. T. Park – "Interframe Coding Using Two-Stage Variable Block-Size Multiresolution Motion Estimation and Wavelet Decomposition" – *IEEE Transactions on CSVT*, vol. 8, no. 04, pp. 399-410, Ag, 1998.

Apêndice A

Provas Matemáticas Referenciadas no Texto

Prova A.1) "Toda seqüência convergente é uma seqüência de Cauchy" [55]

Seja "Uma seqüência $\{x_n\}_{0<n<\infty}$ uma seqüência convergente em ***X***. Então, para qualquer $\varepsilon > 0$ e $\varepsilon \in \Re$, há um número inteiro $N > 0$ tal que $d(x_f, x_n) < \varepsilon, \ \forall n > N$. Também pode-se dizer que $d(x_f, x_m) < \varepsilon, \ \forall m > N$. Logo, pela desigualdade triangular, tem-se:

$$d(x_m, x_n) \le d(x_m, x_f) + d(x_f, x_n)$$
$$d(x_m, x_n) \le 2\varepsilon = \delta$$

Logo, para qualquer $\delta > 0$, há um número inteiro $N > 0$ tal que $d(x_m, x_n) < \delta \ \forall m, n > N$; que é exatamente a definição de Seqüência de Cauchy apresentada na Definição 2 do Capítulo 9.

Prova A.2) Prova do Teorema da Colagem

Suponha $f : \boldsymbol{X} \to \boldsymbol{X}$ um mapeamento contrativo em um espaço métrico completo (***X***,*d*). Seja $x \in \boldsymbol{X}$ e um número real $s \in [0,1)$ o fator de contratividade do mapeamento contrativo *f*. Para $\forall \ m > n \ge 0$:

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

lembrando que $x = f^{o0}(x) = f^{o(n-n)}(x)$.

Pela desigualdade triangular, tem-se que:

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

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

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

$$d\left(f^{on}(x), f^{om}(x)\right) \le \frac{s^n}{1-s} \cdot d\left(x, f(x)\right)$$

Como o atrator é dado por $X_f = f^{o\infty}(x)$ e $x = f^{o0}(x)$:

$$d\left(x_f, x\right) \le \frac{1}{1-s} \cdot d\left(x, f(x)\right).$$

Prova A.3) Prova sobre Contratividade de ω no espaço dos fractais:

"Como ω é uma mapeamento contrativo, então ele é contínuo [55]. Logo, preserva a características topológicas. Assim, tomando-se um subconjunto $B \subset \boldsymbol{X}$ não-vazio, $\omega(B) = \{\omega(x) \mid x \in B\}$ é não vazio. Se B for compacto, então $\omega(B)$ também é compacto [55]. Como o espaço dos fractais $\mathcal{H}(\boldsymbol{X})$ é o espaço dos subconjuntos compactos e não vazios do espaço $\boldsymbol{X}$. Se $\in \mathcal{H}(\boldsymbol{X})$ então $\omega(B) \in \mathcal{H}(\boldsymbol{X})$. Ou seja, a função ω definida desta forma mapeia o espaço $\mathcal{H}(\boldsymbol{X})$ para si mesmo.

Agora para provar que se ω é contínuo em $\boldsymbol{X}$, e´continuo também em $\mathcal{H}(\boldsymbol{X})$, sejam $B, C \in \mathcal{H}(\boldsymbol{X})$, então:

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

$$h(\omega(B),\omega(C)) \le \max\left\{\max_{x\in B}\left\{\min_{y\in C}\{s\cdot d(x,y)\}\right\}, \max_{y\in C}\left\{\min_{x\in B}\{s\cdot d(x,y)\}\right\}\right\} = s\cdot h(B,C) \qquad (34)$$

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

Assim, para determinar se um mapa é contrativo no espaço de fractais $\mathcal{H}(\boldsymbol{X})$, basta determinar a sua contratividade no espaço $\boldsymbol{X}$.

Apêndice B

Códigos-fonte desenvolvidos na pesquisa

Neste apêndice são mostradas as partes principais dos códigos desenvolvidos em Matlab 6.1 durante a pesquisa.

B.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;

%XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
%X                    LOOP PRINCIPAL                          X
%XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

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
```

B.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);
prox_LSP= 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
```

B.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
```

B.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
% XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
% 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
```

B.5 – Função de codificação e decodificação híbrida proposta:

```
function [imgrec_aj, bpp, PSNR] = proc_hibrido(imagem, e_per)

% Codificador
qdtree = [5 5];
bits_SO = [7 9];

% Calculando a TWD
 [C S] = wavedec2(imagem, 3, 'bior4.4');
imgsaida = c2imgsaida(C,S,3);

% Separando as subimagens
PB = imgsaida(        1:S(1,1),          1:S(1,2) )*255/8;
H3 = imgsaida(       [1:S(2,1)], S(2,2)+[1:S(2,2)])*255/8;
V3 = imgsaida(S(2,1)+[1:S(2,1)],        [1:S(2,2)])*255/8;
D3 = imgsaida(S(2,1)+[1:S(2,1)], S(2,2)+[1:S(2,2)])*255/8;
```

```
H2 = imgsaida(      [1:S(3,1)], S(3,2)+[1:S(3,2)])*255/4;
V2 = imgsaida(S(3,1)+[1:S(3,1)],       [1:S(3,2)])*255/4;
D2 = imgsaida(S(3,1)+[1:S(3,1)], S(3,2)+[1:S(3,2)])*255/4;

H1 = imgsaida(      [1:S(4,1)], S(4,2)+[1:S(4,2)])*255/2;
V1 = imgsaida(S(4,1)+[1:S(4,1)],       [1:S(4,2)])*255/2;
D1 = imgsaida(S(4,1)+[1:S(4,1)], S(4,2)+[1:S(4,2)])*255/2;

% Codificando as imagens em fractal
[tc_pb, pt_pb] = fractcoder2(PB, .0001, qdtree, .5, bits_SO);
[tc_h3, pt_h3] = fractcoder2(H3, .0001, qdtree, .5, bits_SO);
[tc_v3, pt_v3] = fractcoder2(V3, .0001, qdtree, .5, bits_SO);
[tc_d3, pt_d3] = fractcoder2(D3, .0001, qdtree, .5, bits_SO);

[tc_h2h3c, pt_h2h3c] = fractcoder_predito(H2, tc_h3, qdtree, bits_SO-4);
[tc_v2v3c, pt_v2v3c] = fractcoder_predito(V2, tc_v3, qdtree, bits_SO-4);
[tc_d2d3c, pt_d2d3c] = fractcoder_predito(D2, tc_d3, qdtree, bits_SO-4);

[tc_h1h3c, pt_h1h3c] = fractcoder_predito(H1, tc_h3, qdtree, bits_SO-4);
[tc_v1v3c, pt_v1v3c] = fractcoder_predito(V1, tc_v3, qdtree, bits_SO-4);
[tc_d1d3c, pt_d1d3c] = fractcoder_predito(D1, tc_d3, qdtree, bits_SO-4);

% Calculando ajustes de matching (eliminando os matching ruins)
[tc_pb_aj, oh_pb] = localiza_erro_pred(tc_pb, PB, e_per*max(max(PB))      );
[tc_h3_aj, oh_h3] = localiza_erro_pred(tc_h3, H3, e_per*max(max(abs(H3))) );
[tc_v3_aj, oh_v3] = localiza_erro_pred(tc_v3, V3, e_per*max(max(abs(V3))) );
[tc_d3_aj, oh_d3] = localiza_erro_pred(tc_d3, D3, e_per*max(max(abs(D3))) );

[tc_h2h3c_aj, oh_h2h3c] = localiza_erro_pred(tc_h2h3c, H2, e_per*max(max(abs(H2))) );
[tc_v2v3c_aj, oh_v2v3c] = localiza_erro_pred(tc_v2v3c, V2, e_per*max(max(abs(V2))) );
[tc_d2d3c_aj, oh_d2d3c] = localiza_erro_pred(tc_d2d3c, D2, e_per*max(max(abs(D2))) );

[tc_h1h3c_aj, oh_h1h3c] = localiza_erro_pred(tc_h1h3c, H1, e_per*max(max(abs(H1))) );
[tc_v1v3c_aj, oh_v1v3c] = localiza_erro_pred(tc_v1v3c, V1, e_per*max(max(abs(V1))) );
[tc_d1d3c_aj, oh_d1d3c] = localiza_erro_pred(tc_d1d3c, D1, e_per*max(max(abs(D1))) );

% Decodificando as imagens em fractal
PBd_aj = fractdecoder_completo(tc_pb_aj, pt_pb, oh_pb, S(1,:), .0001, qdtree, .5);

H3d_aj = fractdecoder_completo(tc_h3_aj, pt_h3, oh_h3, S(2,:), .0001, qdtree, .5);
V3d_aj = fractdecoder_completo(tc_v3_aj, pt_v3, oh_v3, S(2,:), .0001, qdtree, .5);
D3d_aj = fractdecoder_completo(tc_d3_aj, pt_d3, oh_d3, S(2,:), .0001, qdtree, .5);

H2_h3c_aj = fractdecoder_completo(tc_h2h3c_aj, pt_h2h3c, oh_h2h3c, S(3,:), .0001, qdtree, .5);
V2_v3c_aj = fractdecoder_completo(tc_v2v3c_aj, pt_v2v3c, oh_v2v3c, S(3,:), .0001, qdtree, .5);
D2_d3c_aj = fractdecoder_completo(tc_d2d3c_aj, pt_d2d3c, oh_d2d3c, S(3,:), .0001, qdtree, .5);

H1_h3c_aj = fractdecoder_completo(tc_h1h3c_aj, pt_h1h3c, oh_h1h3c, S(4,:), .0001, qdtree, .5);
V1_v3c_aj = fractdecoder_completo(tc_v1v3c_aj, pt_v1v3c, oh_v1v3c, S(4,:), .0001, qdtree, .5);
```

```
D1_d3c_aj = fractdecoder_completo(tc_d1d3c_aj, pt_d1d3c, oh_d1d3c, S(4,:), .0001, qdtree, .5);

% Montando a imagem recuperada
imgwav_aj=[[[PBd_aj H3d_aj;V3d_aj D3d_aj]*2 , H2_h3c_aj; V2_v3c_aj, D2_d3c_aj]*2, ...
                        H1_h3c_aj; V1_v3c_aj, D1_d3c_aj]*2/255;

Crc_aj = imgsaida2c(imgwav_aj, S, 3);
imgrec_aj = sintwaveletv2(Crc_aj, S);
```

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*, somente aguardando data de publicação.

12/2004 **Artigo**: "A New Approach to Reduce Blocking Artifacts 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.
Aprovado pela revista Revista *Ciência e Tecnologia* UNISAL, somente aguardando data de publicação.

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.*