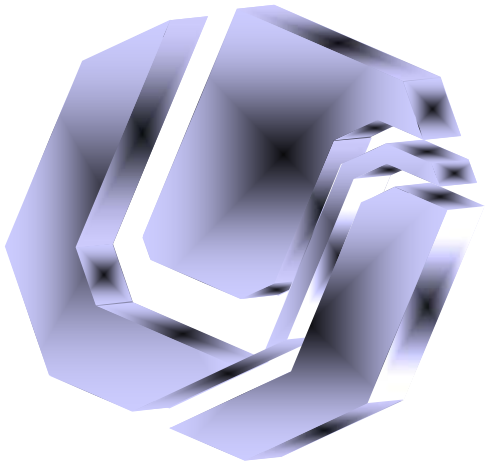
UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA



UM MODELO DE COMPRESSÃO DE IMAGENS DIGITAIS BASEADO EM QUANTIZAÇÃO VETORIAL E TRANSFORMAÇÕES AFINS

JOÃO PAULO IGNÁCIO FERREIRA RIBAS

MARÇO
2008

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

UM MODELO DE COMPRESSÃO DE IMAGENS DIGITAIS BASEADO EM QUANTIZAÇÃO VETORIAL E TRANSFORMAÇÕES AFINS

Tese de doutorado apresentada por João Paulo Ignácio Ferreira Ribas à Universidade Federal de Uberlândia para obtenção do título de Doutor em Ciências, aprovada pela Banca Examinadora:

Prof. Gilberto Arantes Carrijo, Dr., Pós-Doc. (Orientador)

Prof. Cláudio Afonso Fleury, Dr. (CEFET/GO)

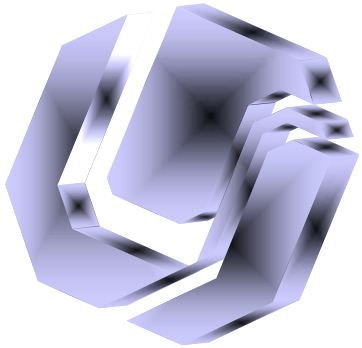
Prof. Sandrerley Ramos Pires, Dr. (Faculdades Alves Faria – ALFA/GO)

Prof. Antonio Cláudio Paschoarelli Veiga, Dr. (UFU)

Profª. Edna Lúcia Flôres, Dra. (UFU)

Uberlândia, 12 de Março de 2008.

UM MODELO DE COMPRESSÃO DE IMAGENS DIGITAIS BASEADO EM QUANTIZAÇÃO VETORIAL E TRANSFORMAÇÕES AFINS



JOÃO PAULO IGNÁCIO FERREIRA RIBAS

Tese de Doutorado apresentada por João Paulo Ignácio Ferreira Ribas à Universidade Federal de Uberlândia como parte dos requisitos para obtenção do título de Doutor em Ciências.

Profº Gilberto Arantes Carrijo Orientador	Profº Darizon Alves de Andrade Coordenador do Curso de Pós-Graduação

A meus pais,

José Ferreira Ribas e Dalva Ignácio Ferreira Ribas.

A meu irmão,

José Antônio Ignácio Ferreira Ribas e família.

"O Senhor é o meu pastor, nada me faltará. Deitar-me faz em verdes pastos, guia-me mansamente a águas tranqüilas. Refrigera a minha alma; guia-me pelas veredas da justiça, por amor do seu nome. Ainda que eu andasse pelo vale da sombra da morte, não temeria mal algum, porque tu estás comigo; a tua vara e o teu cajado me consolam. Preparas uma mesa perante mim na presença dos meus inimigos, unges a minha cabeça com óleo, o meu cálice transborda. Certamente que a bondade e a misericórdia do Senhor me seguirão todos os dias da minha vida; e habitarei na casa do Senhor por longos dias."

Salmo 23

AGRADECIMENTOS

Primeiramente, agradeço a Deus por mais uma vez demonstrar sua grandiosidade, presenteando-me com tão gloriosa oportunidade.

A minha saudosa e querida mãe, Dalva Ignácio Ferreira Ribas (*in memorian*) pela ajuda, em todos os aspectos, mesmo após a sua partida e que, com absoluta certeza, está agora desfrutando daquilo que Deus lhe reservou de melhor, deixando a todos nós um exemplo de vida a ser seguido.

A meu Pai, José Ferreira Ribas, companheiro e amigo inseparável e a meu irmão, José Antônio Ignácio Ferreira Ribas e família, pelo carinho, afeto e, sobretudo, pelo incentivo e apoio dado em todos os momentos de minha vida.

Ao Professor Gilberto Arantes Carrijo, pesquisador de valor inestimável, cuja competência dispensa comentários, por conduzir majestosamente a orientação deste trabalho.

A Universidade Federal de Uberlândia (UFU), instituição respeitada, na qual tenho convivido importantes anos de minha vida.

A Secretaria de Estado de Saúde de Mato Grosso (SES/MT) pelo apoio financeiro e aos amigos e colegas de trabalho pela amizade e companheirismo.

Aos professores componentes da banca examinadora, por participarem do processo de avaliação deste trabalho.

A todos que, direta ou indiretamente, contribuíram para o andamento desta pesquisa.

RESUMO

A compressão fractal é uma técnica emergente de codificação de imagens caracterizada por explorar a auto-similaridade presente nas imagens digitais, que apresenta boa fidelidade entre as imagens original e decodificada, e atinge altas taxas de compressão. Entretanto, apresenta algumas deficiências e por exigir um esforço computacional considerável tem-se utilizado o auxílio de uma ou mais técnicas para suprir essas necessidades. Este trabalho apresenta um modelo de codificação que combina a codificação fractal e a quantização vetorial (VQ), além de minimizar o tempo gasto na escolha da transformação geométrica (isometria), importante etapa da codificação fractal, por ser realizada no domínio da freqüência pelo produto interno da Transformada Discreta Cosseno (DCT). O algoritmo Linde_Buzo_Gray (LBG) é utilizado para designar um *codebook* genérico que substitui o *domain-pool* tradicional de um codificador fractal. O resultado é um codificador híbrido com melhor desempenho que os codificadores fractais puros, que preserva boa qualidade visual da imagem reconstruída e atinge altas taxas de compressão.

Palavras Chave: Fractal, Quantização Vetorial, DCT.

ABSTRACT

The fractal compression is an emerging digital image coding technique which explores the self-similarity present in digital images, showing good fidelity between the original image and the reconstructed image, achieving high compression rates. However it has some weaknesses and because it demands a considerable computational complexity commonly is used the assistance of one or more techniques to meet those needs. This research presents a model which combines fractal coding and vector quantization (VQ). In addition, the time spent in choosing geometric transformation (isometry), which is an important step of fractal coding, is minimized by being made in the frequency domain by the DCT (Discrete Cosine Transform) inner product. The LBG (Linde_Buzo_Gray) algorithm is used to designate a generic codebook that replaces the traditional domain-pool of a fractal coder. The result is a hybrid coder with better performance than the pure fractal coders that preserve the visual quality of the reconstructed image and reaches high compression rates.

Keywords: Fractal, Vector Quantization, DCT.

UM MODELO DE COMPRESSÃO DE IMAGENS DIGITAIS BASEADO EM QUANTIZAÇÃO VETORIAL E TRANSFORMAÇÕES AFINS

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

2-D	Bidimensional
ANSI	American National Standards Institute
AVQ	Address Vector Quantization
bpp	Bits por pixel
BTC	Block Truncation Coding
CCITT	International Telegraph and Telephone Consultative Committee
CR	Compression Rate
dB	Decibel
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DPCM	Differential Pulse Code Modulation
DWT	Discrete Wavelet Transform
FCVQ	Fine-Coarse Vector Quantization
G3	Group 3
G4	Group 4
HVS	Human Visual System
IEC	International Electrotechnical Comission
IFS	Iterated Function System
ISO	International Organization for Standardization
ITU	International Telecommunication Union

JBIG	Joint Bit-Level Image Group
JPEG	Joint Photographic Experts Group
KLT	Karhunen-Loeve Transform
LBG	Linde_Buzo_Gray
LOCOI	Low Complexity Losless Compression for Image
LVQ	Lattice Vector Quantization
LZ	Lempel-Ziv
LZW	Lempel-Ziv -Welch
MSE	Mean Square Error
MSVQ	Multistage Vector Quantization
PCM	Pulse Code Modulation
PIFS	Partitioned Iterated Function System
PSNR	Peak Signal-to-Noise Ratio
PVQ	Pyramid Vector Quantization
RLE	Run Lenght Encoding
ROI	Region of Interest
SGVQ	Shape-Gain Vector Quantization
SNR	Signal-to-Noise Ratio
TCVQ	Trellis-coded Vector Quantization
TSVQ	Tree-Structured Vector Quantization
VQ	Vector Quantization

CAPÍTULO I

INTRODUÇÃO

1.1- Apresentação

O processamento digital de imagens (PDI) é uma área que atrai muitas pesquisas atualmente. Diversos avanços têm se presenciado nos últimos anos devido, principalmente, ao rápido desenvolvimento tecnológico cada vez mais acentuado. Muitas áreas do conhecimento, tais como biologia, medicina, astronomia, engenharia, aplicações industriais, etc., utilizam aplicações baseadas em conceitos e técnicas de PDI, cujos resultados visam, de uma forma geral, a interpretação humana ou a utilização em visão computacional.

A compressão de imagens tornou-se imprescindível em muitas aplicações de PDI, podendo ser livre de perdas ou não, dependendo do ramo de atuação. Com o surgimento de novos serviços de comunicação digital pela Internet ou para comunicações multimídia, como teleconferências e televisão digital, o desenvolvimento de pesquisas na área de compressão de imagens e vídeo tem aumentado substancialmente, sendo um dos assuntos mais abordados em processamento de imagens, contribuindo para o aparecimento de vários padrões e aplicações emergentes, incluindo: gerenciamento eletrônico de documentos (GED), geoprocessamento, sensoriamento remoto, cartografia e muitas outras.

A compressão de imagens por fractais é uma técnica relativamente recente e inovadora, porém ainda apresenta algumas deficiências, sendo a principal delas o elevado

tempo de processamento exigido na etapa de codificação, devido a esse problema, a grande maioria das pesquisas relacionadas a esse assunto envolve estudos visando acelerar este processo. A partir disso, muitos modelos híbridos envolvendo a codificação fractal de imagens têm sido propostos, sempre buscando explorar as melhores características das técnicas envolvidas e com o objetivo de desenvolver codificadores que proporcionem altas taxas de compressão, mantendo a qualidade da imagem reconstruída e com um desempenho satisfatório. Dessa forma, esse trabalho propõe um modelo de codificação híbrido baseado em duas técnicas bastante eficientes, a codificação por fractais e a quantização vetorial. Neste trabalho, os termos "compressão de imagens" e "codificação de imagens" são sinônimos, pois na grande maioria das aplicações que envolvem a codificação de imagens resulta-se um processo de compressão, seja ele com ou sem perdas.

1.2- Referencial Teórico

O princípio da codificação fractal está embasado pelo surgimento da geometria fractal, desenvolvida por Bernoit Mandelbrot em 1975 [1]. Em seguida iniciou-se uma série de estudos referentes ao assunto, onde se destaca a obra de Michael Barnsley [2] que também foi um dos primeiros a propor a codificação de imagens por fractais [3].

Arnaud Jacquin [4, 5], por sua vez desenvolveu o primeiro codificador de imagens digitais, dando início a uma série de pesquisas buscando desenvolver codificadores fractais mais eficientes, dentre os quais se destaca o codificador de Fisher [6, 7].

Após a publicação dos trabalhos de Fisher muitos estudos têm sido desenvolvidos, a grande maioria propondo modelos híbridos de codificação, quase todos visando melhorar o

desempenho em relação aos codificadores fractais puros. Alguns exemplos desses modelos híbridos são apresentados em Davoine, Antonini e Chasser [8], Thao [9], Kim e Park [10], Hamzaoui e Saupe [11] e Iano, Silva e Cruz [12].

A quantização vetorial (VQ), cujos conceitos vieram da quantização escalar, tornou-se uma técnica bastante explorada na codificação de voz e imagens, tendo como referência clássica em processamento de imagens os trabalhos publicados por Gray [13] e Linde, Buzo e Gray [14], sendo constantemente referenciados em muitas pesquisas atuais [15]. Alguns trabalhos publicados utilizam a combinação da codificação fractal e da quantização vetorial (VQ), como pode ser observado em Davoine, Antonini e Chasser [8], Kim e Park [10] e Hamzaoui e Saupe [11], sendo este assunto o principal foco desta tese.

1.3- Objetivos deste Trabalho

O principal objetivo deste trabalho é o desenvolvimento de um modelo de compressão de imagens digitais, utilizando as técnicas de codificação fractal e quantização vetorial (VQ), referenciado nesta tese como modelo de codificação Fractal-VQ. Durante a pesquisa é preciso investigar minuciosamente as características das técnicas envolvidas, buscando explorar o que cada uma tem de melhor para obter os melhores resultados possíveis.

Para alcançar o objetivo geral proposto, os seguintes objetivos específicos foram estabelecidos, cuja composição conduz ao objetivo principal:

- Levantar o estado da arte referente à compressão de imagens digitais, buscando conhecer as principais técnicas e padrões disponíveis atualmente;
- Fazer um estudo da teoria dos fractais, incluindo a fundamentação matemática da geometria fractal e da codificação fractal de imagens;

- Desenvolver um codificador fractal de imagens aplicando os conhecimentos obtidos no item anterior;
- Estudar a fundamentação teórica da quantização vetorial e implementar um quantizador vetorial de imagens;
- Construir um novo codificador híbrido Fractal-VQ e apresentar resultados, bem como comparações que demonstrem a eficiência do modelo proposto, e;

1.4- Estrutura da Tese

Esta tese está dividida em oito capítulos, sendo o Capítulo I a introdução geral ao tema, contendo o referencial teórico levantado, objetivos e estrutura da tese.

O segundo capítulo traz uma introdução à compressão de imagens digitais, apresentando uma visão geral das técnicas e padrões mais conhecidos.

Nos Capítulos III e IV é apresentada a fundamentação teórica da geometria fractal e da codificação fractal de imagens, abordando as principais definições e teoremas referentes ao assunto. Ainda no Capítulo IV é descrito o funcionamento de um sistema tradicional de compressão fractal de imagens baseado nos conceitos apresentados no Capítulo III, também são mostrados alguns resultados práticos resultantes da implementação do codificador em questão.

No Capítulo V é discutida a técnica de quantização vetorial (VQ), incluindo a fundamentação teórica e a implementação de um quantizador de imagens cujo *codebook* é gerado pelo algoritmo Linde_Buzo_Gray (LBG), sendo considerado muito eficiente desde a sua criação.

O Capítulo VI descreve todos os detalhes do modelo de codificação proposto neste trabalho, este modelo é um esquema híbrido que combina codificação fractal e quantização vetorial (VQ), motivo pelo qual é referenciado como modelo de codificação Fractal-VQ. Os Resultados e comparações são apresentados no Capítulo VII.

O Capítulo VIII contém as conclusões gerais e as contribuições deste trabalho, bem como sugestões para futuros trabalhos.

1.5- Considerações Finais deste Capítulo

Neste capítulo foi apresentada a contextualização do assunto a ser discutido neste trabalho. O referencial teórico abordado na seção 1.2 cita alguns dos principais trabalhos e autores mencionados constantemente nas pesquisas que englobam a codificação fractal e a quantização vetorial (VQ), visto que o objetivo principal desta pesquisa é o desenvolvimento de um novo modelo de compressão de imagens que associe essas duas técnicas. Esta tese está estruturada em oito capítulos, conforme descritos na seção 1.4, sendo que o próximo, o Capítulo II, descreverá o estado da arte referente à compressão de imagens digitais, abordando as técnicas e padrões mais conhecidos.

CAPÍTULO II

COMPRESSÃO DE IMAGENS DIGITAIS

2.1- Introdução

Em virtude do progressivo aumento das aplicações que necessitam de tecnologias de compressão para minimizar o espaço de armazenamento e a largura de banda necessária para a transmissão, a compressão de dados tornou-se importante foco de pesquisas nos últimos anos e é atualmente um assunto bastante explorado em processamento digital de sinais (PDS). Inúmeros trabalhos têm sido apresentados e estudos demonstram que, apesar de encontrar-se em um avançado estágio de desenvolvimento, as pesquisas referentes à compressão de dados ainda têm muito a contribuir.

A compressão de imagens visa reduzir o número de bits necessários para representar uma imagem digital, tal procedimento pode ser realizado de duas maneiras: 1) sem perda, com a eliminação da informação redundante, podendo-se posteriormente recuperar-se totalmente a imagem original ou 2) com perda, descartando-se, além da informação redundante, a informação julgada insignificante, que depende muito da aplicação em questão.

Em uma imagem digital pode-se encontrar três tipos de redundâncias: de código, interpixel e psicovisual [16]. A redundância de código refere-se à probabilidade de ocorrência de um determinado valor de pixel (tom de cinza) na imagem, de forma que pixels com maiores taxas de ocorrência são codificados com menos bits e vice-versa. A redundância

interpixel (ou espacial) permite prever o valor de um determinado pixel da imagem pelos valores de seus pixels vizinhos. A redundância psicovisual está relacionada com a percepção visual humana, sendo assim, a informação que tem menor importância visual em uma imagem pode ser descartada, levando obrigatoriamente a uma compressão com perdas.

Diversas técnicas de compressão de imagens estão disponíveis na literatura e muitos padrões vêm sendo desenvolvidos ao longo dos anos.

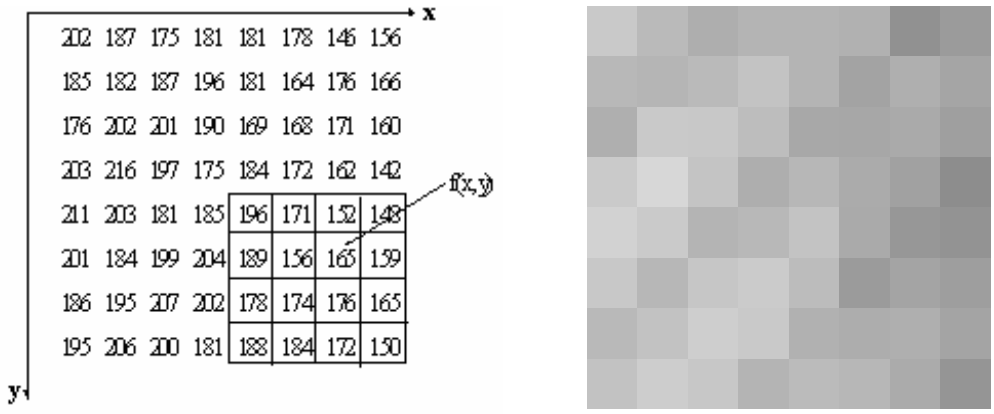
Este capítulo apresenta uma visão geral das mais conhecidas técnicas de compressão de imagens e de alguns padrões de codificação presentes na atualidade.

2.2- Representação de Imagens Digitais

Uma imagem monocromática é um sinal analógico bidimensional *f(x,y)* processado pelo Sistema Visual Humano (HVS), onde *x* e *y* indicam as coordenadas espaciais e o valor de *f* em um determinado ponto *(x,y)* é proporcional ao brilho (ou nível de cinza) da imagem naquele ponto [16].

Nas aplicações computacionais, para o processamento, armazenamento e transmissão, é preciso converter a imagem da forma analógica para digital. Assim, a digitalização de uma imagem *f(x,y)* é realizada discretizando-a tanto em coordenadas espaciais (amostragem) quanto em amplitude (quantização em níveis de cinza). Neste trabalho só serão tratadas imagens em níveis de cinza.

A representação matemática de uma imagem digital qualquer é uma matriz de números inteiros, onde cada elemento *f(x,y)* representa o nível de cinza naquele ponto *(x,y)*. As Figuras 2.1(a) e 2.1(b) mostram uma representação matricial de uma imagem digital 8 x 8 e a imagem digitalizada com 256 níveis de cinza, respectivamente.



(a) (b)

Figura 2.1 – (a) Representação matricial de uma imagem digitalizada 8 x 8 com 256 níveis de cinza; (b) imagem digitalizada.

2.3- Técnicas de Compressão de Imagens

As técnicas de compressão de imagens são divididas em dois grandes grupos: com perdas (*lossy*) e sem perdas (*lossless*) [17]. A escolha depende se é preciso ou não obter uma réplica exata da imagem original a partir da imagem reconstruída.

A compressão sem perdas (lossless) permite a recuperação exata da imagem após o processo de descompressão. Em certos tipos de aplicações não é permitido ocorrer perdas de informação, como em imagens de satélite ou imagens médicas, onde a diminuição da qualidade pode comprometer o diagnóstico. A compressão com perdas é a mais usual, como na Internet, por exemplo, é utilizada para remover, além da informação redundante, dados insignificantes à percepção visual humana e permite apenas uma recuperação aproximada da imagem original.

2.4- Técnicas de Compressão sem Perdas

A codificação sem perdas de imagens pode ser vista como um procedimento de dois estágios: descorrelação e codificação por entropia [18]. O primeiro estágio, utilizado para remover a redundância espacial ou redundância inter-pixel, é realizado utilizando-se técnicas de descorrelação, tais como: Run-lenght Encoding (RLE), codificação por plano de bits, técnicas de transformada e outras. O segundo estágio é responsável pela remoção da redundância de código, que pode ser realizada por alguma técnica de codificação por entropia, já que são usadas técnicas estatísticas para eliminar ou minimizar a redundância, dentre elas destacam-se: codificação de Huffman, codificação aritmética e codificação de Lempel-Ziv-Welch (LZW), esta última também pode ser utilizada na remoção de redundância espacial [18].

A Figura 2.2 mostra um resumo das técnicas de compressão de imagens sem perdas.

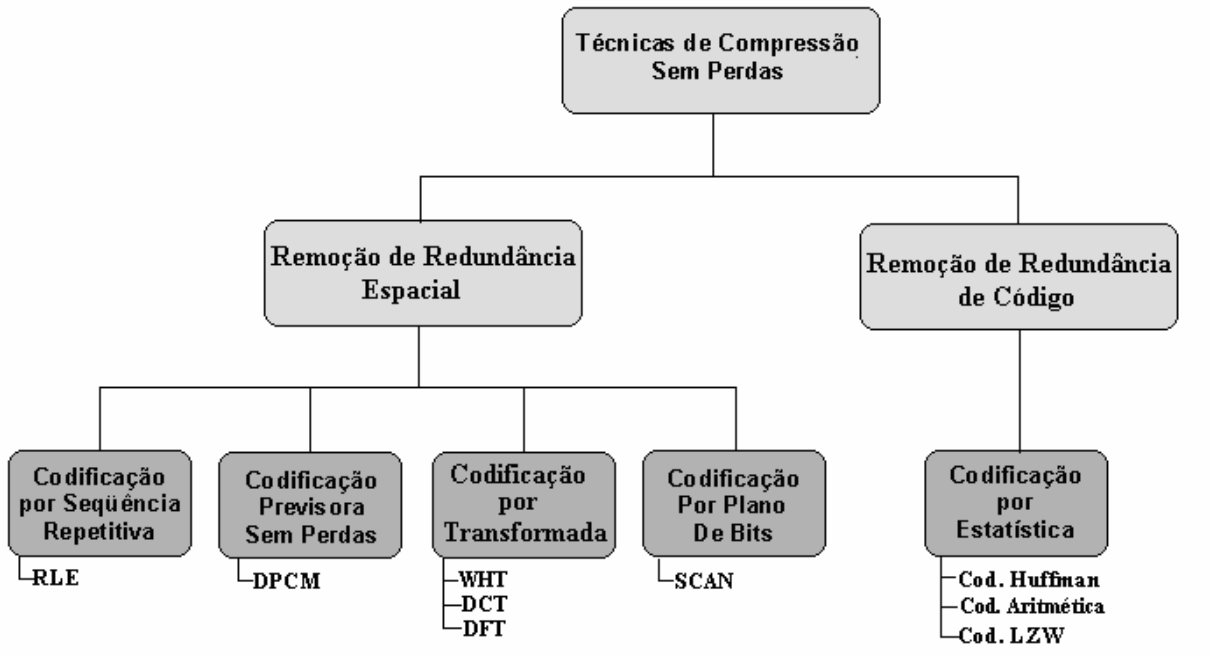


Figura 2.2 – Técnicas de compressão de imagens sem perdas.

2.4.1- Codificação de Huffman

Em 1952, Huffman inventou a primeira técnica de compressão sem perdas, nesta técnica é descoberta a frequência relativa de cada símbolo (pixel), que é codificado em função da sua probabilidade de ocorrência. Dessa forma, os símbolos que ocorrem mais frequentemente são codificados com um menor número de bits, enquanto os que ocorrem com menor frequência são codificados com um número de bits relativamente maior [16].

A codificação de Huffman é considerada ótima, desde que os símbolos da fonte sejam codificados um por vez. Contudo, ela não resulta em uma alta taxa de compressão, visto que os símbolos não possuem um padrão fixo de probabilidades e isso causa uma expansão dos dados codificados.

Muitos padrões de codificação de imagens utilizam inicialmente técnicas com perdas e, posteriormente, codificação de Huffman na finalização do processo.

2.4.2- Codificação Aritmética

Como a codificação de Huffman, a codificação aritmética é uma técnica estatística. Só que ao invés de se codificar cada símbolo individualmente, é atribuída uma palavra de código aritmético (em um intervalo de números reais entre 0 e 1) a uma sequência de símbolos. Dessa forma, a correlação entre os pixels vizinhos pode ser explorada [17].

A codificação aritmética também é utilizada no processamento final de diversas aplicações e padrões de compressão de imagens.

2.4.3- Codificação de Lempel-Ziv (LZ)

Jacob Ziv e Abraham Lempel publicaram em 1977 um algoritmo para compressão de dados sequenciais, que ficou conhecido como LZ-1 ou LZ-77. [17]

O método LZ-1 baseia-se em armazenar, em um dicionário ou tabela, as sequências de símbolos (pixels) que ocorrem com mais frequência. Cada sequência de símbolos da imagem original é representada no dicionário apenas pelo seu índice, que contém a posição de início e o tamanho da sequência. Dessa forma, as longas sequências de símbolos podem ser codificadas em códigos menores, resultando em uma compressão sem perdas.

Os formatos de arquivos de imagens *Tagged Image File Format* (TIFF) e *Graphical Interchange Format* (GIF), muito comuns na Internet, utilizam Lempel-Ziv como técnica de codifição [17].

2.4.4- Codificação por Seqüência Repetitiva – *Run Length Encoding* (RLE)

A RLE provavelmente é a técnica de compressão de dados mais conhecida. Entretanto ela proporciona uma baixa taxa de compressão. Basicamente, é realizada a substituição de uma sequência de símbolos idênticos (pixels) por símbolos de menores tamanhos. O maior problema da RLE é que se a informação contiver uma proporção significativa de símbolos diferentes entre si, o arquivo comprimido tende a ser tão grande quanto o arquivo de entrada.

A RLE é utilizada usualmente, nos sistemas de compressão, em conjunto com uma técnica de compressão com perdas, como foi dito anteriormente. O formato BMP de imagens digitais também utiliza o algoritmo RLE.

Atualmente, na compressão de mapas e imagens médicas com grande quantidade de regiões homogêneas a RLE resulta em uma boa compressão [19].

2.4.5- *Differential Pulse Code Modulation* (DPCM) sem Perdas

A Codificação Previsora (*Predictive Coding*) é uma técnica de compressão de imagens que se baseia na suposição que cada pixel pode ser obtido pela combinação linear dos pixels vizinhos [17]. Em vez de transmitir uma imagem, a diferença entre uma previsão da imagem (valor previsto) e a imagem original é codificada e transmitida. Essa diferença é chamada de erro de previsão. Na decodificação, esse erro é adicionado ao valor previsto da amostra.

Os algoritmos de previsão unidimensional exploram a correlação entre os pixels adjacentes em uma mesma linha. Outros algoritmos mais complexos utilizam a correlação linha a linha e quadro a quadro, que são chamados de previsão bidimensional e tridimensional, respectivamente.

A *Differencial Pulse Code Modulation* (DPCM) é o tipo mais comum de codificação previsora sem perdas.

Na *DPCM* sem perdas o valor de cada pixel da imagem original, com exceção dos limites, é previsto pelo o valor dos pixels vizinhos com a finalidade de se obter uma imagem prevista. Com isso, pela diferença entre os valores do pixel original e o pixel previsto, obtém-se a *imagem* diferencial ou *imagem* residual, que é menor do que a imagem original e os valores dos pixels são menos dinâmicos.

Finalmente, a imagem diferencial é codificada com bastante eficiência pela codificação de Huffman (*Huffman Encoding*), descrita anteriormente no subitem 2.4.1.

2.4.6- Codificação por Transformada

Os sistemas de codificação baseados em transformadas são geralmente enquadrados como técnicas de compressão com perdas (*lossy*) pelo fato de que a maioria delas, como a Transformada Discreta de Fourier (DFT) e a Transformada Discreta Cosseno (DCT), produzem coeficientes reais ou complexos, necessitando serem quantizados após a transformação para serem codificados. Entretanto, a transformada de Walsh-Hadamard (WHT) pode ser aplicada diretamente na codificação de sinais sem perda, pois os coeficientes resultantes são frações binárias, não sendo preciso quantizá-los antes da codificação [18]. A codificação por transformada (com perdas) será abordada novamente na seção 2.5.3.

2.4.7- Codificação por Plano de Bits

Na codificação por plano de bits uma imagem (níveis de cinza ou colorida) é decomposta em várias imagens binárias. Por exemplo, uma imagem com 256 níveis de cinza (8 bits) pode ser representada por 8 planos de 1 bit com as mesmas dimensões da imagem original. A partir daí, cada imagem binária pode ser codificada utilizando-se uma técnica sem perdas, como por exemplo, RLE. [17]

Na grande maioria das imagens, os pixels são correlacionados. Com isso, o valor entre os pixels vizinhos pouco diferem. Assim, nos planos binários, os valores dos bits de uma determinada região da imagem são similares, contribuindo para a obtenção de uma boa compressão.

A linguagem SCAN, utilizada em processamento de imagens, adota a metodologia da codificação por plano de bits, essa linguagem quando ela é utilizada em conjunto com a DPCM, RLE ou Huffman [18], tem apresentado bons resultados.

Uma comparação entre alguns padrões de compressão sem perdas de imagens pode ser encontrada em Arps e Truong [20].

2.5- Técnicas de Compressão com Perdas

As técnicas de compressão com perdas são muito variadas atualmente, sendo as mais usuais, principalmente pela disseminação da Internet nos últimos anos. A Figura 2.3 apresenta uma taxonomia das técnicas de compressão com perdas, adaptada de Subramanya [17].

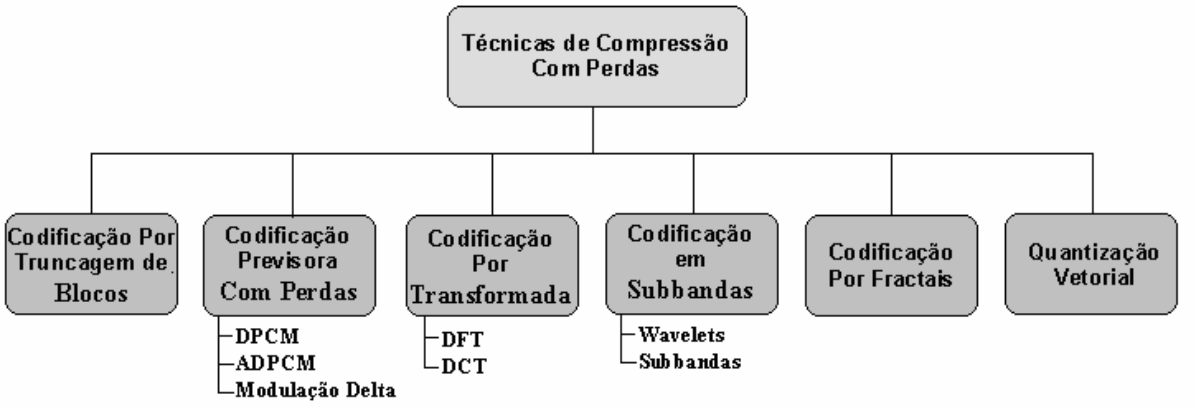


Figura 2.3 – Técnicas de compressão de imagens com perdas.

Na prática, a maioria dos sistemas de compressão com perdas é híbrida [17], inicialmente eles utilizam a combinação de técnicas com perdas para a codificação, como por exemplo: codificação por transformada e codificação previsora; codificação em sub-bandas e codificação por transformada; codificação previsora e quantização vetorial; e muitas outras.

Posteriormente, no estágio final da codificação, adotam-se as técnicas de codificação sem perdas, como Huffman ou codificação aritmética. A Figura 2.4, adaptada de Subramanya [17], mostra um sistema de compressão com perdas.

Figura 2.4 – Sistema de compressão de imagens com perdas.

Na Figura 2.4, no primeiro passo é realizada uma das seguintes operações: 1) previsão da imagem original, na qual o valor de cada pixel é previsto baseado nos pixels vizinhos, derivando uma imagem diferencial, que é a diferença entre a imagem prevista e a imagem original. 2) transformada da imagem original, que é obtida aplicando-se uma transformada matemática (DFT, DCT, Hadamard e muitas outras) que transporta a imagem do domínio espacial para o domínio da frequência. 3) decomposição da imagem original em diferentes componentes, no domínio da frequência. Em cada caso (1, 2 ou 3) existe uma operação inversa no receptor, que efetuada sobre a nova representação, resulta na imagem original (sem perdas).

No segundo passo da Figura 2.4 é realizada a quantização, que desconsidera a informação insignificante para a percepção visual humana.

No terceiro passo aplica-se a codificação por entropia, onde uma técnica de compressão sem perdas é utilizada para finalizar o processo de compressão.

A decodificação é similar, basta executar o processo inverso: 1) a decodificação por entropia é aplicada na imagem comprimida para obter-se a imagem quantizada. 2) efetua-se a

desquantização e finalmente 3) a transformação inversa para reconstruir a imagem, que é uma aproximação da imagem original.

Com relação ao desempenho, as técnicas de compressão com perdas são analisadas mediante três considerações, que são as mais importantes:

1) Taxa de compressão (CR), que é obtida por [17]:

$$CR = \frac{tamanho\ da\ imagem\ original}{tamanho\ da\ imagem\ comprimida} \tag{2.1}$$

2) Relação Sinal-Ruído de Pico (PSNR), que é obtida por [21]:

$$PSNR = 10 \cdot \log \frac{(2^n - 1)^2}{MSE}\ dB \tag{2.2}$$

onde:

n – número de bits por pixel;

MSE – erro médio quadrático, que é calculado por [21]:

$$MSE = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (x_{i,j} - \hat{x}_{i,j})^2 \tag{2.3}$$

M,N – número de pixels;

$x_{i,j}$ - valor do pixel original na posição i,j;

$\hat{x}_{i,j}$ – valor do pixel reconstruído na posição i,j;

3) A velocidade de codificação e decodificação;

2.5.1- Codificação por Truncagem de Blocos (BTC)

Na codificação por truncagem de bloco (BTC) a imagem é dividida em blocos de pixels não sobrepostos, originalmente 4 x 4 pixels, que são processados separadamente. Para cada bloco são calculados e codificados o valor médio (limiar) dos pixels e o desvio padrão (σ) [22].

A quantização de cada bloco é realizada substituindo-se os valores dos pixels que estão abaixo do limiar por 0-bit e o resto dos pixels assumem valor 1-bit, resultando em um mapa de bits do quadro.

Sabendo-se o valor médio dos pixels (limiar) , o desvio padrão (σ) , a quantidade de 0's e 1's é possível obter-se o bloco reconstruído, com perdas.

A Figura 2.5, adaptada de Subramanya [17] ilustra uma visão geral da BTC.

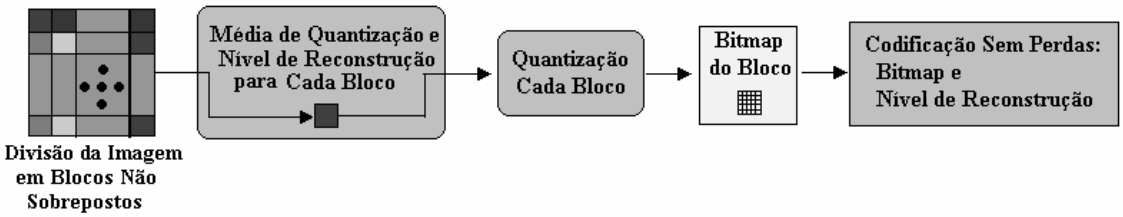


Figura 2.5 – Codificação por truncagem de blocos (BTC).

2.5.2- *Differential Pulse Code Modulation* (DPCM) com Perdas

A DPCM é uma técnica de codificação previsora (predictive coding) popular. A DPCM com perdas é muito parecida com a DPCM sem perdas, descrita no ítem 2.4.5. A grande diferença é que na DPCM sem perdas o valor de cada pixel da imagem original é

previsto pelo valor dos pixels vizinhos, enquanto que na DPCM com perdas o valor dos pixels são previstos pelos valores reconstruídos dos píxels vizinhos, como pode ser visto na Figura 2.6, adaptada de Subramanya [17].

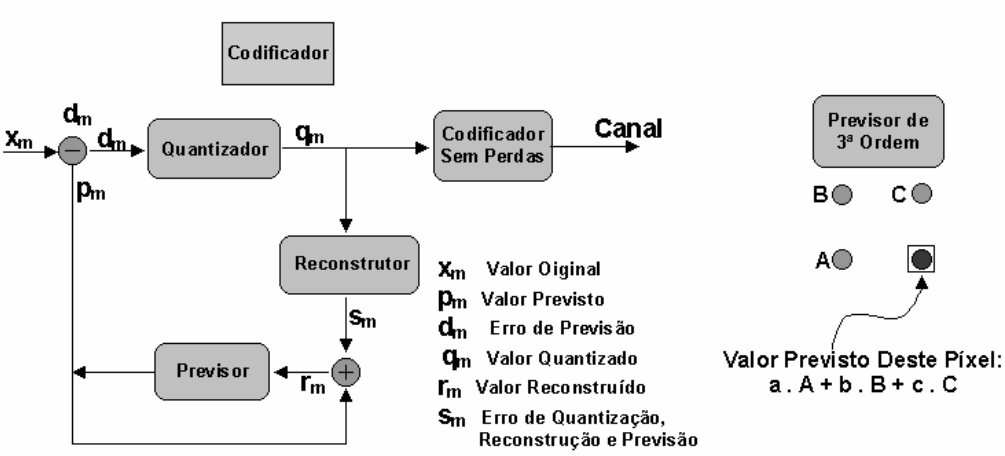


Figura 2.6 – DPCM com perdas.

Na DPCM da Figura 2.6 foi usado um previsor de terceira ordem, onde três valores previstos são utilizados para prever cada pixel.

2.5.3- Codificação por Transformada

As técnicas discutidas até aqui neste capítulo efetuam as operações diretamente nos pixels de uma imagem, portanto estão no domínio espacial. A codificação por transformada é uma técnica no domínio da frequência.

A transformada é uma ferramenta matemática que possibilita transportar uma matriz de pixels (imagem) no domínio espacial para o domínio da frequência. Dessa forma, a maior

parte da energia do sinal fica concentrada em poucos coeficientes da matriz transformada, que são então quantizados e codificados [16].

Existem várias transformadas de imagem, cada qual com suas características e propriedades. Dentre elas pode-se citar: a Transformada de Karhunen-Loeve (KLT), a Transformada de Fourier (DFT), a Transformada Cosseno (DCT), a Transformada de Hadamard, a Transformada Wavelet (DWT) e muitas outras. As transformadas de imagem podem ser estudadas com bastante detalhes em Jain [21].

A codificação por transformada, ilustrada na Figura 2.7, consiste em subdividir uma imagem de dimensões *N x N* em sub-imagens com dimensões *n x n* e aplicar a transformada em cada bloco isoladamente. Em seguida, armazena-se os coeficientes que apresentam maior concentração da energia, que são quantizados e posteriormente codificados utilizando alguma técnica de codificação por entropia (sem perdas) [17].

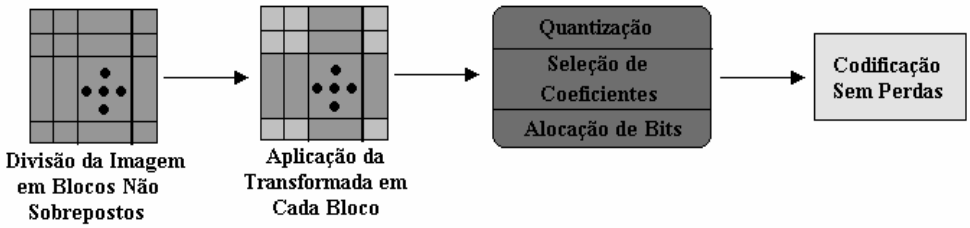


Figura 2.7 – Codificação por transformada.

Na decodificação, efetua-se o processo inverso. Para reconstruir a imagem a partir da matriz de coeficientes da transformada, utiliza-se a transformada inversa.

Atualmente, o padrão mais popular de compressão de imagens é o *Joint Photographic Experts Group* (JPEG), que utiliza a transformada cosseno (DCT) em blocos de dimensões 8 x 8. Uma estratégia para compressão de imagens médicas baseada na DCT é proposta em Wu e Chuan [23].

2.5.4- Codificação em Sub-bandas

A codificação em sub-bandas baseia-se no princípio da decomposição do sinal de entrada (imagem) em múltiplas sub-bandas utilizando-se um conjunto de filtros passa faixa, onde cada sub-banda sofre um processo de decimação, que é responsável pela definição da largura da sub-banda, que é então codificada separadamente [24].

A vantagem da utilização do esquema de codificação em sub-bandas é que com uma alocação de bits apropriada nas diferentes sub-bandas, de acordo com um critério percentual estabelecido (decimador), o número de níveis de quantização e a variação do erro de reconstrução podem ser controlados separadamente em cada sub-banda [25].

Na decodificação, as sub-bandas codificadas são decodificadas, interpoladas e filtradas pelo banco de filtros correspondente e, finalmente, são somadas para a reconstrução do sinal original.

A Figura 2.8, adaptada de Subramanya [17], ilustra uma visão do processo de codificação em sub-bandas.

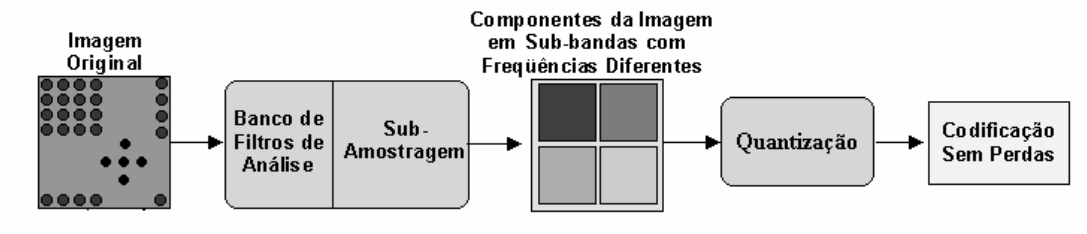


Figura 2.8 – Codificação em sub-bandas.

2.5.5- Codificação por Transformada Wavelet (DWT)

Atualmente a transformada Wavelet vem apresentado-se como uma poderosa ferramenta para compressão de imagens, visto que ela permite representar uma imagem em multirresoluções e com as altas frequências podendo serem facilmente localizadas. A DWT não requer a subdivisão da matriz de pixels (imagem) em blocos, como ocorre na codificação usando outras transformadas. Os algoritmos que utilizam a DWT demonstram um desempenho tão satisfatório ou superior quanto o JPEG [26].

A Figura 2.9, adaptada de Zhang, Morgan e Greenwood [26], apresenta o diagrama em blocos do codificador de imagens que utiliza a transformada wavelet.

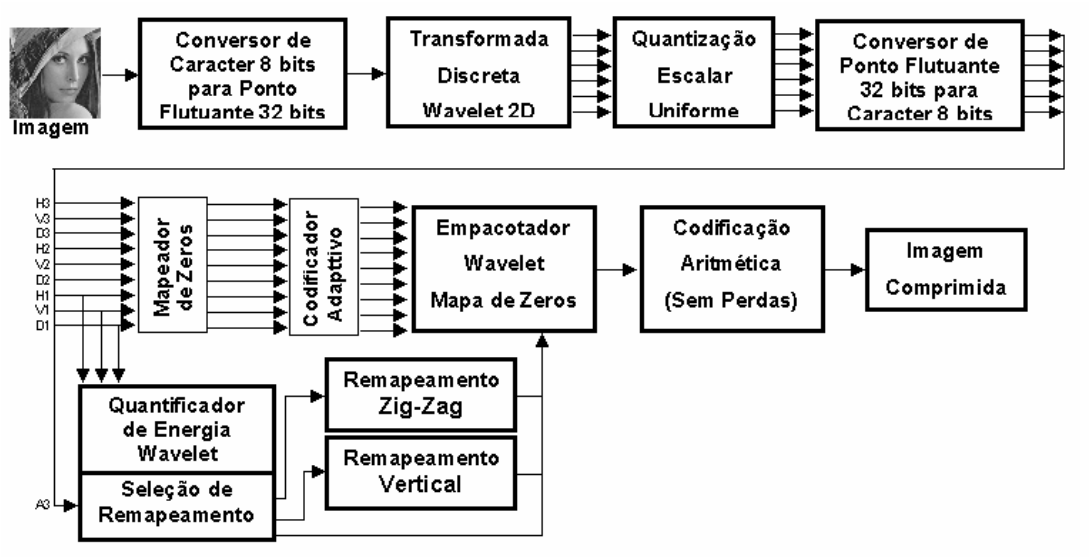


Figura 2.9 – Codificação wavelet.

Para uma melhor precisão da DWT, a imagem original é convertida para ponto flutuante (*float*) 32-bit. A DWT é baseada no banco de filtros 10/18 biortogonal. Como exemplo, uma DWT de três níveis foi escolhida, que resulta em 10 arquivos sub-bandas. Após

a transformação é utilizado um quantizador escalar uniforme, que produz um grande número de zeros nas sub-bandas de alta freqüência, especialmente quando a taxa de bits é baixa (menor que 1 bit/píxel). Na estrutura hierárquica da DWT, se existirem zeros em alto nível de frequência (baixa resolução), os coeficientes correspondentes ao baixo nível (alta resolução) também estarão próximos de zero, desde que representem a mesma localização espacial. O mapa de zeros é designado para explorar esta auto-similaridade [26].

Na Figura 2.9, os três níveis da DWT, usada no exemplo, especifica os arquivos para as margens horizontal (H), vertical (V) e diagonal (D) respectivamente. ‘A’ representa os componentes de baixa freqüência. Os números após essas letras indicam o nível da DWT.

2.5.6- Quantização Vetorial (VQ)

A quantização vetorial é um método de compressão de imagens no qual o princípio básico é dividir o fluxo de dados em blocos chamados vetores. No caso de uma imagem, o vetor é um pequeno bloco retangular ou quadrado de pixels de tamanho fixo chamado vetor imagem.

Define-se como dicionário de codificação (*codebook*) um conjunto de vetores padrões de tamanho fixo, que está disponível na codificação e na decodificação.

Para cada vetor imagem de entrada, o dicionário de codificação (*codebook*) é consultado e o vetor padrão mais próximo é selecionado. Com isso, apenas o índice do vetor padrão é transferido. Assim, a imagem passa a ser representada por uma sequência de índices que podem ser codificados entropicamente.

A Figura 2.10 mostra uma representação da quantização vetorial [17].

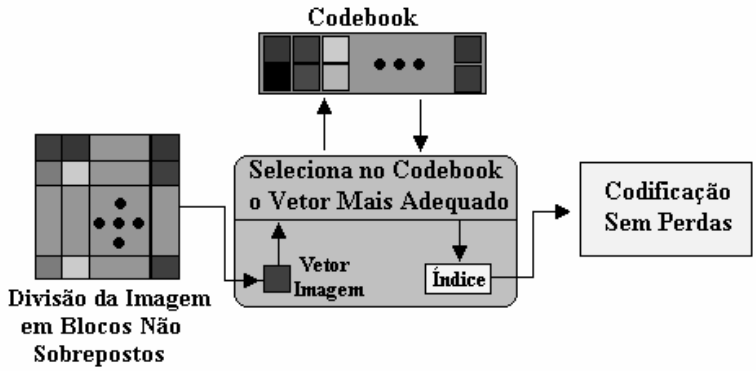


Figura 2.10 – Quantização vetorial.

2.5.7- Codificação por Fractais

Na codificação com base em fractais é realizada a decomposição da imagem em segmentos chamados fractais, que são definidos seguindo critérios de processamento de imagens tais como: separação de cores, detecção de margens e análise espectral e de textura.

A Figura 2.11, adaptada de Subramanya [17] ilustra a codificação de imagens por fractais. Cada segmento da imagem original é procurado em uma biblioteca de fractais, cujo conteúdo são códigos chamados *Iterated Function System* (IFS) [27]. Sendo assim, é determinado um conjunto de códigos para representar essa imagem, de forma que quando os códigos IFS são aplicados aos blocos de imagens é possível reconstruir uma imagem bem próxima da original.

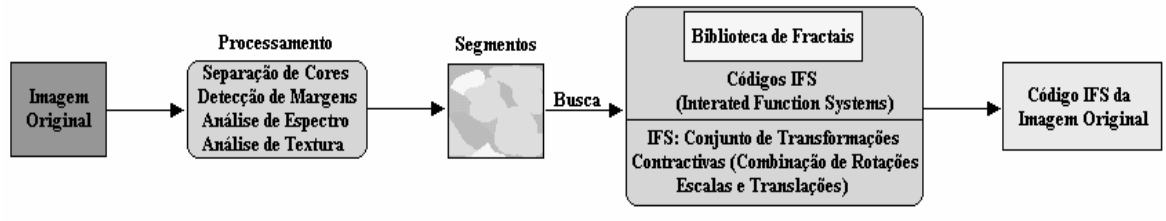


Figura 2.11 – Codificação por fractais.

A codificação por fractais apresenta uma boa eficiência na compressão de imagens que possuem boa regularidade e auto-similaridade [17].

2.6- Padrões de Compressão de Imagens

Devido à existência de diversas plataformas, sistemas operacionais e aplicações, os padrões de compressão de imagens foram desenvolvidos para facilitar a interoperabilidade das técnicas existentes. A maioria desses padrões utilizam a codificação híbrida, basicamente desenvolvida com a utilização de algumas das técnicas mencionadas anteriormente neste capítulo.

As maiores organizações de padronização de compressão de imagens existentes são: a *International Organization for Standardzation/International Electrotechnical Comission* (ISO/IEC) e a *International Telecommunication Union - Telecommunication Standardzation Sector* (ITU-T), antigamente chamada *International Telegraph and Telephone Consultative Committee* (CCITT), que são responsáveis pelo desenvolvimento da grande maioria dos atuais padrões de compressão de imagens. Elas, em conjunto, tratam tanto da compressão de imagens binária como imagens em tons contínuos (monocromáticas e coloridas). Um documento de padronização oficial desenvolvido pela CCITT é chamado *Recommendation*, enquanto que da ISO/IEC é chamado de *International Standard*, elas estão disponíveis no *American National Standards Institute* (ANSI), com sede nos Estados Unidos da América [20]. Os mais conhecidos grupos de padrões de compressão de imagens são: JPEG para imagens contínuas [28] ; G3 (*Group 3*), G4 (*Group 4*) e JBIG (*Joint Bi-Level Image Group*) para imagens binárias. A transmissão de fac-símiles é a aplicação mais comum que usa compressão de imagens binárias.

Atualmente, muitos padrões de compressão estão disponíveis no mercado, o JPEG-LS e JPEG-2000 são os mais recentes padrões de compressão de imagens em tons contínuos desenvolvidos pelo grupo JPEG e disponibilizados pela ISO/ITU [18]. A palavra "*joint*" no nome JPEG significa que o grupo deve se reportar tanto para a ISO como para o ITU [29].

O JPEG-LS é um modelo de compressão sem perdas baseado no algoritmo conhecido como *Low Complexity Losless Compression for Image* (LOCOI), caracterizado por apresentar um bom equilíbrio entre complexidade e eficiência [18]. O JPEG-2000 é um novo padrão de compressão de imagens baseado na tecnologia wavelet, ele possui muitas características inovadoras, tais como: performance superior, compressão com perdas ou sem perdas, codificação por região de interesse (ROI), escalabilidade espacial e de relação-sinal-ruído (SNR), dentre outras. Para maiores detalhes sobre o padrão JPEG-2000 consulte Yang e Bourbakis [18], Douglas e Bouzerdoum [29] e Bojkovicé e Milovanovié [30].

2.7- Considerações Finais deste Capítulo

Neste capítulo foram abordadas algumas das mais conhecidas técnicas de compressão de imagens digitais, classificadas em dois grupos, com perdas e sem perdas, as quais representam a essência de muitas aplicações e padrões disponíveis.

O próximo capítulo apresenta a fundamentação teórica da geometria fractal, mostrando as principais definições e teoremas referentes a esse assunto.

CAPÍTULO III

GEOMETRIA FRACTAL

3.1- Introdução

A palavra 'Fractal', que deriva do Latim ('fractus', o adjetivo de 'frangere', que significa 'quebrar') foi apresentada por Benoit Mandelbrot em 1975, quando surgiam os primeiros estudos relacionados à geometria fractal, atualmente é considerada por muitos como um ramo da matemática, possuindo propriedades particulares e demonstrações já estabelecidas.

A geometria clássica, também referenciada como geometria euclidiana, utiliza os métodos apropriados para o estudo de objetos espaciais que são descritos pelos elementos básicos: pontos, retas, segmentos de retas, planos, curvas, ângulos e superfícies. Entretanto, muitos padrões não podem ser descritos perfeitamente somente com a utilização da geometria clássica, sendo necessária a utilização de geometrias alternativas com diferentes estruturas descritivas. Neste contexto, surge a geometria fractal, que fornece métodos para a modelagem e a descrição de objetos que apresentam irregularidades e alto grau de complexidade.

A geometria fractal possui a facilidade de melhor se adaptar à representação de objetos naturais tais como: nuvens, montanhas, o litoral, criaturas aquáticas e muitas outras formas mais complexas com inúmeras irregularidades, contrastando com a geometria euclidiana que se preocupa com a modelagem de objetos criados pelo homem, caracterizados por possuírem formas perfeitas, simétricas e regulares. Dessa forma, a geometria euclidiana exprime-se por

equações e fórmulas enquanto que a geometria fractal utiliza-se de algoritmos e fórmulas iterativas, geralmente sendo necessário o auxílio do computador como ferramenta indispensável.

Este capítulo apresenta a fundamentação teórica de geometria fractal e de codificação fractal de imagens, mostrando as principais definições e teoremas referentes a esse assunto, também são mostrados alguns exemplos clássicos de fractais. Finalmente, são realizadas as considerações finais deste capítulo.

3.2- Definição de Fractal

Nesta seção serão apresentados alguns conceitos de fundamental importância relacionados à definição de fractal.

3.2.1- Espaço Métrico

Um espaço métrico (X, d) é um espaço, ou um conjunto de pontos, X juntamente com uma função real (aplicação) $d:X \times X \rightarrow IR$, que mede a distância entre pares de pontos x e y em X, onde d possui as seguintes propriedades [3]:

(i) $d(x,y) = d(y,x), \ \forall x, y \in X$

(ii) $0 < d(x,y) < \infty, \ \forall x, y \in X, x \neq y$

(iii) $d(x, x) = 0, \ \forall \ x \in X$ (3.1)

(iv) $d(x,y) \leq d(x,z) + d(z,y), \ \forall x, y, z \in X$

Então, d é uma métrica sobre o espaço X.

3.2.2- Transformação, Mapa ou Mapeamento

Seja X um espaço, uma transformação (mapa ou mapeamento) sobre X é uma função $f:X \to X$. Se S é um subconjunto de X, ou seja $S \subset X$, então $f(S) = \{f(x) : x \in S\}$. A função f é um-para-um se $x,y \in X$ e $f(x) = f(y)$ para $x = y$. Se f é uma função um-para-um com $f(X) = X$, então f é chamada inversível e, neste caso, é possível definir a transformação $f^{-1} : X \to X$, denominada inversa de f, por $f^{-1}(y) = x$, onde $x \in X$ é o único ponto em que $y = f(x)$. [3]

3.2.3- Iterações Sucessivas

Seja $f : X \to X$ uma transformação sobre um espaço X, as iterações sucessivas de f são transformações $f^{on} : X \to X$ definidas por [3]:

$$f^{o0}(x) = x$$

$$f^{o1}(x) = f(x)$$

$$f^{o(n+1)}(x) = (f \circ f^{on})(x) = f(f^{on}(x)), \text{ para } n = 0, 1 \ , 2, 3, \ldots \quad (3.2)$$

Se f é inversível, então as iterações reversas de f são transformações $f^{o(-m)}(x) : X \to X$ definidas por:

$$f^{o(-1)}(x) = f^{-1}(x)$$

$$f^{o(-m)}(x) = (f^{om})^{-1}(x), \text{ para } m = 0, 1, 2, 3, \ldots \quad (3.3)$$

3.2.4- Transformações Afins

Transformações afins em *IR* são transformações $f : IR \rightarrow IR$ da seguinte forma:

$$f(x) = a\,x + b, \ \forall x \in IR, \tag{3.4}$$

onde *a* e *b* são constantes reais.

Considere o intervalo $I = [0, 1]$, $f(I)$ é um novo intervalo de forma que o ponto *0* em *I* é movido para *b* em $f(I)$ e $f(I)$ localiza-se à direita ou à esquerda de *b*, conforme *a* seja positivo ou negativo respectivamente, dessa forma $f(I) = [b-a, b+a]$. [3]

A ação de uma transformação afim sobre *IR* pode ser descrita da seguinte forma: uma linha $f(x)$, representada pela Equação (3.4), é alongada se $|a| > 1$ ou contraída se $|a| < 1$ e se $a < 0$ ela é rotacionada 180^0 em relação à origem. A linha é transladada de uma quantidade b para a esquerda ou para a direita, conforme $b < 0$ ou $b > 0$, respectivamente.

3.2.5- Sequência de Cauchy

Uma seqüência $\{x_n\}_{n=1}^{\infty}$ de pontos em um espaço métrico (X,d) é denominada seqüência de Cauchy se para qualquer número $\varepsilon > 0$ existe um inteiro $N > 0$ de forma que [3]:

$$d(x_n, x_m) < \varepsilon \text{ para todo } n, m > N.$$

3.2.6- Limite da Sequência

Uma seqüência $\{x_n\}_{n=1}^{\infty}$ de pontos em um espaço métrico (X,d) é dita convergente a um ponto $x \in X$ se para qualquer número $\varepsilon > 0$ existe um intero $N > 0$ de forma que [3]:

$d(x_n, x) < \varepsilon$ para todo $n > N$.

O ponto $x \in X$ para o qual a seqüência converge é chamado limite da seqüência, com a seguinte notação: $x = \lim_{n\to\infty} x_n$.

Seja $S \subset X$ um subconjunto de um espaço métrico *(X,d)*, um ponto $x \in X$ é chamado ponto limite de *S* se existe uma seqüência $\{x_n\}_{n=1}^{\infty}$ de pontos, $x_n \in S\{x\}$, de forma que $\lim_{n\to\infty} x_n = x$.

Teorema: Se uma seqüência de pontos $\{x_n\}_{n=1}^{\infty}$ em um espaço métrico *(X,d)* converge para um ponto $x \in X$, então $\{x_n\}_{n=1}^{\infty}$ é uma seqüência de Cauchy [3].

3.2.7- Espaço Métrico Completo

Um espaço métrico *(X,d)* é dito completo se toda seqüência de Cauchy $\{x_n\}_{n=1}^{\infty}$ em *X* tem limite $x \in X$ [3]. Por exemplo, o espaço euclidiano IR^2 é um espaço métrico completo.

3.2.8- Subespaço Compacto

Seja $S \subset X$ um subconjunto de um espaço métrico *(X,d)*, *S* é chamado compacto se toda seqüência infinita $\{x_n\}_{n=1}^{\infty}$ em S contém uma subseqüência que possui um limite em *S* [3].

3.2.9- Subespaço Limitado e Totalmente Limitado

Seja $S \subset X$ um subconjunto de um espaço métrico (X,d), S é limitado se existe um ponto $a \in X$ e um número $R > 0$ de forma que $d(a, x) < R$, $\forall x \in X$.

Diz-se que S é totalmente limitado se para cada $\varepsilon > 0$ existe um conjunto finito de pontos $\{y_1, y_2, \ldots, y_n\} \subset S$ tal que $\forall x \in X$, $d(x, yi) < \varepsilon$ para algum $y_i \in \{y_1, y_2, \ldots, y_n\}$ [3].

3.2.10- Subespaço Aberto, Subespaço Fechado e Ponto Limite

Seja $S \subset X$ um subconjunto de um espaço métrico (X,d), então:

Diz-se que S é aberto, se para cada $x \in S$ existe um $\varepsilon < 0$ tal que $B(x,\varepsilon) = \{y \in X: d(x,y) \leq \varepsilon\} \subset$ S. Essa definição é equivalente a: S é aberto se ele não é fechado [3].

O subespaço S é dito fechado se ele contiver todos os seus pontos limites [3].

Um ponto $x \in X$ é um ponto limite de S se para todo número $\varepsilon > 0$, $B(x, \varepsilon)$ contém um ponto em $X \setminus S$ e um ponto em S. O conjunto de todos os pontos limites de S é chamado de limite de S, denotado por ∂S [3].

Teorema: Seja (X,d) um espaço métrico completo e $S \subset X$ um subespaço de X, então S é compacto se, e somente se, é fechado e totalmente limitado.

3.2.11- Ponto Interior

Seja $S \subset X$ um subconjunto de um espaço métrico (X,d), um ponto $x \in X$ é um ponto interior de S se existe um número $\varepsilon > 0$, tal que $B(x, \varepsilon) \subset S$. O conjunto de todos os pontos interiores de S é chamado de interior de S e é denotado por S^0 [3].

3.2.12- Ponto Fixo

Seja $f : X \to X$ uma transformação sobre um espaço e $x_f \in X$ um ponto, tal que $f(x_f) = x_f$, este ponto é chamado ponto fixo da transformação [3].

3.2.13- Contração e Fator de Contração

Uma transformação afim $f: X \to X$ sobre um espaço métrico (X,d) é chamada contração se existe uma constante s, onde $0 \leq s < 1$, tal que

$$d(f(x), f(y)) \leq s.d(x,y) \ , \ \forall x,y \in X.$$

O número real s é chamado fator de contração para a transformação f [3].

Teorema do mapeamento contrativo (Teorema de Banach) [3]: Considere a contração $f: X \to X$ sobre um espaço métrico (X,d), então f possui exatamente um ponto fixo $x_f \in X$ e,

ainda, para qualquer ponto $x \in X$, a seqüência $\{f^{on}(x): n = 0, 1, 2, ...\}$ converge para x_f. Isto é: $\lim_{n\to\infty} f^{on}(x) = x_f$, para cada $x \in X$.

3.2.14- Espaço de Hausdorff

Seja (X,d) um espaço métrico completo, o espaço de Hausdorff, denotado por $H(X)$, é o espaço cujos pontos são subconjuntos compactos de X não vazios [3].

3.2.15- Métrica de Hausdorff

Seja (X,d) um espaço métrico completo, onde $x \in X$ é um ponto e $A,B,C \in H(X)$ são subconjuntos compactos de X não vazios, define-se [3]:

(i) A distância do ponto x ao conjunto A é obtida por:

$$d(x,A) = min\{d(x,y), y \in A\}; \quad (3.5)$$

(ii) A distância entre os dois subconjuntos A e B é encontrada por:

$$d(A,B) = max\{d(x,B), x \in A\}; \quad (3.6)$$

(iii) A distância entre a união de dois subconjuntos A, B e um subconjunto C é encontrada por:

$$d(A \cup B, C) = d(A,C) \vee d(B, C), \quad (3.7)$$

onde $x \vee y$ representa o máximo entre x e y;

(iv) A distância de Hausdorff entre dois subconjuntos A e B $\in H(X)$ é definida por:

$$h(A,B) = max\{d(A,B), d(B,A)\},$$

também pode-se denotar $h(A,B) = d(A,B) \vee d(B, A)$. (3.8)

h é chamada métrica de Hausdorff em H.

Teorema: Seja (X,d) um espaço métrico completo, então $(H(X),h)$ também é um espaço métrico completo.

O espaço métrico de Hausdorff $(H(X),h)$ é o espaço métrico onde os fractais existem.

3.2.16- Contração no Espaço de Hausdorff

Lema: Seja $w : X \rightarrow X$ uma contração com fator de contração s no espaço métrico (X,d), então [3]:

(i) w é contínua;

(ii) w aplica $H(X)$ nele próprio;

(iii) $w : H(X) \rightarrow H(X)$ definida por $w(B) = \{w(x), x \in B\}, \forall B \in H(X)$ é uma contração em $(H(X), h(d))$ com fator de contração s.

Pode-se também utilizar um método para combinar uma seqüência de contrações em $(H(X),h)$ com a finalidade de se produzir novas contrações em $(H(X),h)$.

Lema: Seja (X,d) um espaço métrico e $\{w_n, n = 1, 2, ..., N\}$ um conjunto de contrações em $(H(X),h)$ e seja s_n o conjunto de fatores de contração para cada n em w_n, define-se $W : H(X) \rightarrow H(X)$ por:

$$W(B) = w_1(B) \cup w_2(B) \cup \ldots \cup w_N(B) = \bigcup_{n=1}^{N} w_n(B)\,,\ \forall\ B \in H(X)\,. \tag{3.9}$$

Então W é uma contração em $H(X)$ com fator de contração $s = max\ \{s_n : n= 1, 2, ..., N\}$.

3.2.17- Sistema de Funções Iterativas (IFS)

Um sistema de funções iterativas consiste de um espaço métrico completo (X,d) juntamente com um conjunto finito de contrações $w_n : X \to X$, com respectivos fatores de contração s_n ($n = 1, 2, ..., N$). A notação para um IFS é $\{X; w_n, n = 1, 2, ..., N\}$ com fator de contração $s = max\{s_n, n = 1, 2, ... , N\}$ [3].

Teorema sobre IFS: Seja $\{X; w_n, n = 1, 2, ..., N\}$ um IFS com fator de contração s, então a transformação $W : H(X) \to H(X)$ definida por

$$W(B) = \bigcup_{n=1}^{N} w_n(B)\,,\ \forall\ B \in H(X), \tag{3.10}$$

é uma contração no espaço métrico completo $(H(X),h(d))$ com fator de contração s e sendo assim:

(i) $h(W(B),W(C)) \leq s\ .\ h(B,C),\ \forall\ B,C \in H(X).$ (3.11)

(ii) W possui um único ponto fixo A que obedece

$$A = W(A) = \bigcup_{n=1}^{N} w_n(A)\,,\ \forall\ A \in H(X) \tag{3.12}$$

e que a seqüência $\left\{W^{on}(B)\right\}_{n=1}^{\infty}$ converge para A, dessa forma, A é obtido por

$$A = \lim_{n\to\infty} W^{on}(B)\,,\ \forall\ B \in H(X). \tag{3.13}$$

Ao subconjunto $A \in X$ denomina-se fractal ou atrator. Ou seja, um fractal é o ponto fixo de um IFS em um espaço métrico de Hausdorff.

A notação matricial utilizada para representar as contrações de um IFS em IR^2 pode ser da seguinte forma [3]:

$$w_n(x) = w_n\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_n & b_n \\ c_n & d_n \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e_n \\ f_n \end{bmatrix} = A_n x + t_n \ . \tag{3.14}$$

Para facilitar a representação de dados, o conjunto de contrações de um IFS pode ser obtido pela Tabela 3.1, chamada código IFS (*IFS code*).

Tabela 3.1- Código IFS

w	***a***	***b***	***c***	***d***	***e***	***f***
i	a_i	b_i	c_i	d_i	e_i	f_i

3.3- Dimensão do Fractal

A dimensão de um fractal está intimamente relacionada com a sua quantidade de irregularidades, ela representa o grau de ocupação do fractal no espaço, ao contrário da geometria euclidiana, a dimensão de um fractal não é um número inteiro.

Para melhor entender o conceito de dimensão fractal, primeiramente será apresentada a definição de bola fechada [31]: Seja (X,d) um espaço métrico completo e $A \in X$ um conjunto não vazio, seja $\varepsilon > 0$ um número, então $B(x, \varepsilon)$ é definido como uma bola fechada de raio ε

centrada em um ponto de *x*. Utiliza-se a notação *N(A, ε)* para o menor número de bolas fechadas de raio ε para cobrir o conjunto *A*. Como exemplo, seja o conjunto *A* o fractal conhecido como triangulo de Sierpinsky, a Figura 3.1 ilustra o conceito de bola fechada .

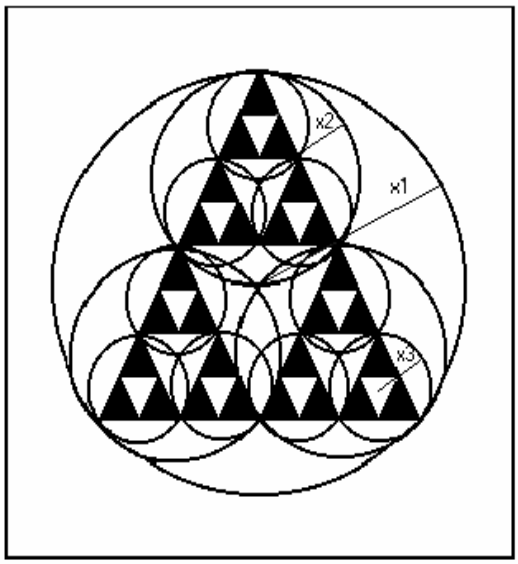


Figura 3.1- Triângulo de Sierpinsk após 3 iterações.

Na Figura 3.1 pode-se verificar que para $\varepsilon = x_1$, $N(A, \varepsilon) = 1$; $\varepsilon = x_2$, $N(A, \varepsilon) = 3$ e para $\varepsilon = x_1$, $N(A, \varepsilon) = 9$

Seja $A \in H(X)$ um subconjunto compacto não vazio de *X* , onde *(X,d)* é um espaço métrico completo e ainda, para cada $\varepsilon > 0$, seja *N(A, ε)* o menor número de bolas fechadas de raio ε necessárias para cobrir *A*, a dimensão fractal do conjunto *A*, denotada por *D(A)* é definida por [2]:

$$D(A) = \lim_{\varepsilon \to 0} \left[\frac{\ln(N(A,\varepsilon)}{\ln\left(\frac{1}{\varepsilon}\right)} \right], \tag{3.15}$$

se existir.

Um método bastante conhecido e utilizado para estimar a dimensão fractal de objetos e imagens é conhecido como teorema da contagem de cubos (*box-counting theorem*) [2]. Sua aplicação consiste em sobrepor uma malha de quadrados ou caixas, de modo a obter o número de quadrados necessários para cobri-la. Seja $A \in H(R^m)$ (espaço de Hausdorff definido no espaço dos números reais de dimensão *m*) um subconjunto fechado e limitado no R^m, cobrindo-se R^m com quadrados (caixas) de arestas = $\frac{1}{2^n}$ e seja *N(A)* o número de cubos que tem interseção não nula com *A*, então se:

$$D(A) = \lim_{n \to 0} \left[\frac{\ln(N(A))}{\ln(2^n)} \right] \tag{3.16}$$

existir, *A* possui dimensão fractal igual a *D*. Para exemplificar será novamente utilizado o fractal triângulo de Sierpinsk, ou seja, A = {triângulo de Sierpinsk com centro (0,0), (0,1) e (1,0)} $\in H(R^2)$, como ilustrado na Figura 3.2.

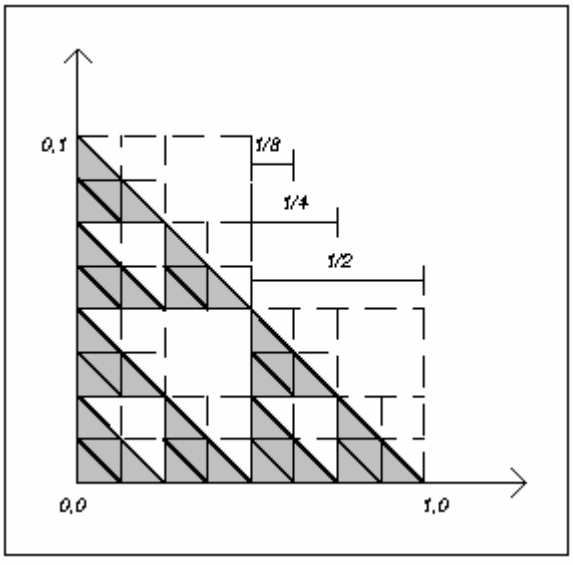


Figura 3.2- Triângulo de Sierpinsk coberto com quadrados.

Se $n = 1$, lado do cubo = $\frac{1}{2^1} = \frac{1}{2}$ e $N_1(A) = 3 = 3^1$;

Se $n = 2$, lado do cubo = $\frac{1}{2^2} = \frac{1}{4}$ e $N_1(A) = 9 = 3^2$;

Se $n = 3$, lado do cubo = $\frac{1}{2^3} = \frac{1}{8}$ e $N_1(A) = 27 = 3^3$;

⋮

Ou seja, $N_n(A) = 3^n$, logo a dimensão fractal do triângulo de Sierpinsk é obtida por:

$$D(A) = \lim_{n\to\infty} \frac{\ln 3^n}{\ln 2^n} = \lim_{n\to\infty} \frac{n\ln 3}{n\ln 2} = \frac{\ln 3}{\ln 2} \approx 1{,}585$$

Um outro conceito de dimensão fractal é chamada dimensão fractal de Hausdorff-Besicovitch ou dimensão por autosemelhança, que é um valor D tal que $N = \left(\frac{1}{s}\right)^D$, onde s é o fator de contração do IFS e N é o número total de transformações (contrações) que constituem o IFS, dessa forma [1]:

$$D = \frac{\log N}{\log\left(\frac{1}{s}\right)} \tag{3.17}$$

Na seção seguinte serão apresentados alguns exemplos de fractais clássicos na literatura, cujas dimensões serão calculadas utilizando-se a dimensão de Hausdorff-Besicovitch.

3.4- Exemplos de Fractais

Na maioria das referências bibliográficas sobre fractais encontram-se exemplos clássicos de IFS's, bem como dos fractais por eles gerados. Nesta seção serão apresentados alguns deles.

3.4.1- Curva de Koch

A curva de Koch é um fractal gerado por um IFS: $\{IR^2; w_1, w_2, w_3, w_4\}$, com fator de contração $s = \frac{1}{3}$, ilustrado na Figura 3.3.

Figura 3.3- Curva de Koch.

O processo para a construção da curva de Koch inicia-se com uma linha simples unidimensional que não ocupa espaço. Após várias iterações o contorno da curva se extende por uma área finita, ocupando um espaço mais do que unidimensional, mas que não chega ser bidimensional. A dimensão dessa curva é obtida por:

$$D = \frac{\log 4}{\log\left(\frac{1}{1/3}\right)} = \frac{\log 4}{\log 3} = 1{,}2618$$

3.4.2- Triângulo de Sierpinsky

Considere o seguinte IFS: $\{IR^2; w_1, w_2, w_3\}$, em que IR^2 é o espaço métrico completo com métrica de Hausdorff e w_1, w_2, w_3 o conjunto de contrações, onde:

$$w_1(x,y) = \left(\frac{1}{2}x, \frac{1}{2}y\right) \qquad w_2(x,y) = \left(\frac{1}{2}x, \frac{1}{2}y\right) + \left(\frac{1}{2}, 0\right) \qquad w_3(x,y) = \left(\frac{1}{2}x, \frac{1}{2}y\right) + \left(\frac{1}{4}, \frac{\sqrt{3}}{4}\right)$$

Utilizando-se a notação matricial apresentada na seção 3.2.17 deste capítulo, tem-se:

$$w_1(x) = w_1\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad , \quad w_2(x) = w_2\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0 \end{bmatrix} \quad \text{e}$$

$$w_3(x) = w_3\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.25 \\ 0.43 \end{bmatrix}$$

O código IFS é obtido pela Tabela 3.2.

Tabela 3.2- Um código IFS para gerar o triângulo de Sierpinsk.

w	***a***	***b***	***c***	***d***	***e***	***f***
1	*0.5*	*0*	*0*	*0.5*	*0*	*0*
2	*0.5*	*0*	*0*	*0.5*	*0.5*	*0*
3	*0.5*	*0*	*0*	*0.5*	*0.25*	*0.43*

Então, os fatores de contração são obtidos por: $s_1 = \frac{1}{2}$, $s_2 = \frac{1}{2}$, $s_3 = \frac{1}{2}$ e o fator de contração do IFS é encontrado por $s = max\{s_1, s_2, s_3\} = \frac{1}{2}$. Como o sistema métrico é o IR^2, a iteração pode-se iniciar com qualquer conjunto compacto não vazio de IR^2, como por exemplo, um quadrado cheio. As iterações são obtidas por:

$$B_1 = W(B_0) = w_1(B_0) \cup w_2(B_0) \cup w_3(B_0)$$

$B_2 = W(B_1) = w_1(B_1) \cup w_2(B_1) \cup w_3(B_1) = w_1(w_1(B_0) \cup w_2(B_0) \cup w_3(B_0)) \cup w_2(w_1(B_0) \cup w_2(B_0) \cup w_3(B_0)) \cup w_3(w_1(B_0) \cup w_2(B_0) \cup w_3(B_0)) = (w_1(w_1(B_0)) \cup w_1(w_2(B_0)) \cup w_1(w_3(B_0)) \cup (w_2(w_1(B_0)) \cup w_2(w_2(B_0)) \cup w_2(w_3(B_0)) \cup (w_3(w_1(B_0)) \cup w_3(w_2(B_0)) \cup w_3(w_3(B_0))$

$B_N = W(B_{N-1}) = w_1(B_{N-1}) \cup w_2(B_{N-1}) \cup w_3(B_{N-1})$.

A Figura 3.4 ilustra as primeiras iterações do IFS para gerar o triângulo de Sierpinsk.

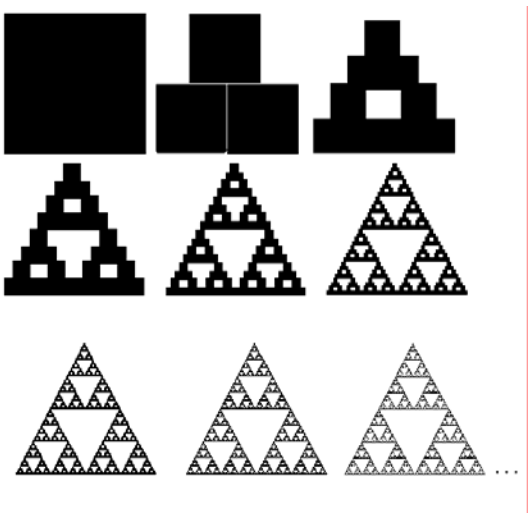


Figura 3.4- Geração do triângulo de Sierpinsk.

A dimensão do triângulo de Sierpinsk é obtida por:

$$D = \frac{\log 3}{\log\left(\frac{1}{1/2}\right)} = \frac{\log 3}{\log 2} = 1{,}585$$

3.4.3- Samambaia de Barnsley

A samambaia de Barnsley é gerada por um IFS: *{IR²; w_1, w_2, w_3}*. Sabe-se da seção 3.2.17 deste capítulo que

$$w_n(x) = w_n\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_n & b_n \\ c_n & d_n \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e_n \\ f_n \end{bmatrix} = A_n x + t_n .$$

Neste exemplo, a matriz A é da forma $A = \begin{bmatrix} r_1 \cos\theta_1 & -r_2 \operatorname{sen}\theta_2 \\ r_1 \operatorname{sen}\theta_1 & r_2 \cos\theta_2 \end{bmatrix}$ e as contrações são obtidas por:

$$w_1\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{3}{3{,}25}\cos(-2) & -\frac{2{,}8}{3{,}25}sin(-3) \\ \frac{3}{3{,}25}sin(-2) & \frac{2{,}8}{3{,}25}\cos(-3) \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} -0{,}5 \\ 0{,}75 \end{pmatrix}$$

$$w_2\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{0{,}95}{3{,}25}\cos(49) & -\frac{1{,}1}{3{,}25}sin(49) \\ \frac{0{,}95}{3{,}25}sin(49) & \frac{1{,}1}{3{,}25}\cos(49) \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 1{,}3 \\ -7 \end{pmatrix}$$

$$w_3\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{1}{3{,}25}\cos(-49) & -\frac{1{,}25}{3{,}25}sin(-54) \\ \frac{1}{3{,}25}sin(-49) & \frac{1{,}25}{3{,}25}\cos(-54) \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 8{,}4 \\ -7{,}95 \end{pmatrix}$$

Após as iterações o IFS produz o atrator conhecido como samambaia de Barnsley. A Figura 3.5 mostra o resultado após 9 iterações.

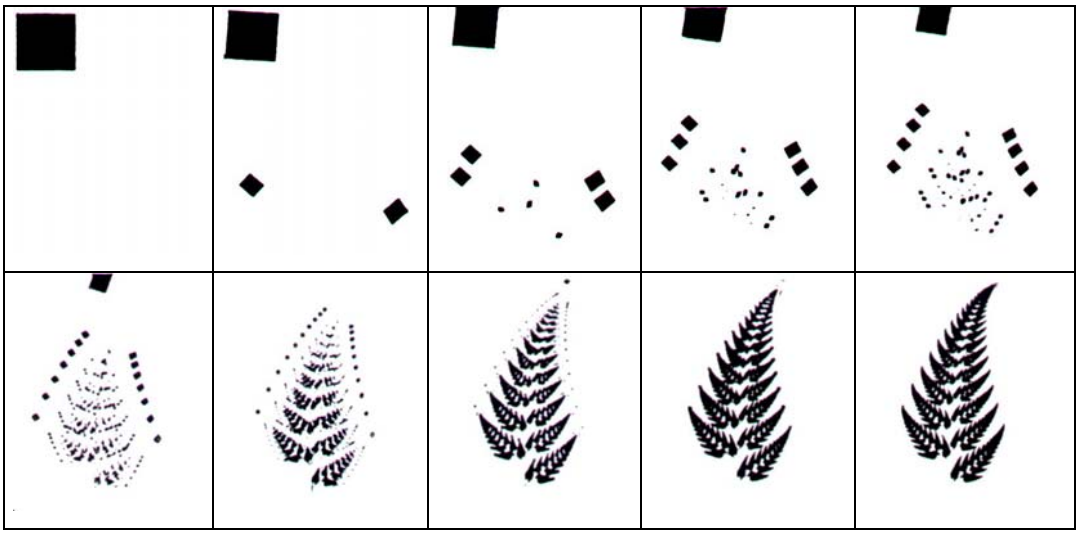


Figura 3.5- Samambaia de Barnsley.

3.4.4- Curva de Peano

A geração da curva de Peano por um IFS: $\{IR^2; w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9\}$ possui o código IFS apresentado na Tabela 3.

Tabela 3.3- Código IFS para gerar a curva de Peano

w	*a*	*b*	*c*	*d*	*e*	*f*
1	*1/3*	*0*	*0*	*1/3*	*0*	*0*
2	*1/3*	*0*	*0*	*-1/3*	*0*	*-1/3*
3	*1/3*	*0*	*0*	*1/3*	*-1/3*	*-1/3*
4	*-1/3*	*0*	*0*	*1/3*	*1/3*	*-2/3*
5	*-1/3*	*0*	*0*	*-1/3*	*-2/3*	*0*
6	*-1/3*	*0*	*0*	*1/3*	*0*	*-1/3*
7	*1/3*	*0*	*0*	*1/3*	*-1/3*	*-1/3*
8	*1/3*	*0*	*0*	*-1/3*	*1/3*	*2/3*
9	*1/3*	*0*	*0*	*1/3*	*-2/3*	*0*

O processo iterativo inicia-se com um segmento de reta unitário, após a primeira iteração o IFS produz nove segmentos de reta com 1/3 do tamanho da reta inicial, posicionados como mostrado na Figura 3.6.

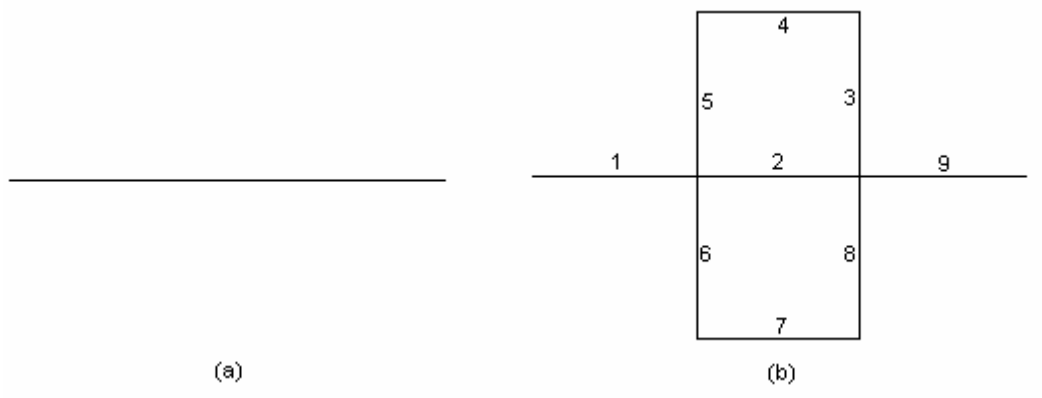


Figura 3.6- (a) Reta unitária; (b) primeira iteração da curva de Peano.

Após cinco iterações obtém-se um resultado como mostrado na Figura 3.7. A dimensão desse atrator é obtida por:

$$D = \frac{\log 9}{\log\left(\frac{1}{1/3}\right)} = \frac{\log 9}{\log 3} = 2$$

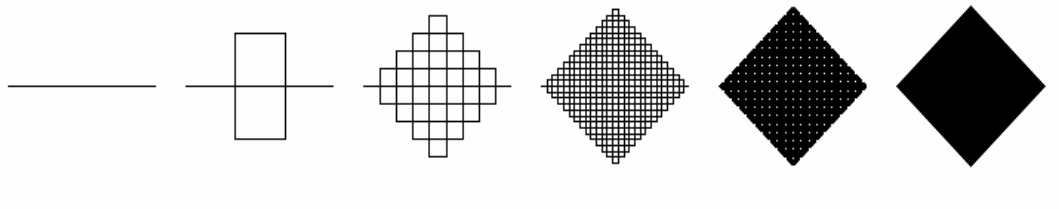


Figura 3.7- Geração da curva de Peano.

3.5- Características dos Fractais

Os fractais possuem algumas características particulares que os diferenciam de outros objetos [1]. A principal delas é a auto-similaridade ou auto-semelhança, que pode ser definida

de duas formas diferentes. Uma delas é a auto-semelhança exata, onde o conjunto total é obtido a partir de cópias fiéis do conjunto inicial. Alguns exemplos para visualizar este conceito foram apresentados na seção 3.4 deste capítulo, onde se pode observar que os fractais gerados são obtidos por sucessivas cópias do objeto inicial. Este tipo de auto-semelhança não existe em objetos naturais, apenas em teoria. Sendo assim, o conceito de auto-similaridade estatística enquadra-se mais adequadamente à modelagem de objetos do mundo real, que refere-se a uma certa semelhança entre os conjuntos menores e o conjunto final, apresentando os mesmos padrões e características em termos gerais, não sendo o atrator um agrupamento de réplicas fiéis do conjunto inicial.

Outra característica importante observada nos fractais é a independência de escala, ou seja, o grau de ampliação não altera o nível de complexidade de detalhamento do atrator, que se confunde com o objeto inicial por apresentarem auto-similaridade.

Os fractais são gerados por funções iterativas, portanto, o grau de detalhamento de um fractal é diretamente proporcional ao número de iterações do IFS, podendo ser ampliado o quanto for necessário. Dessa forma, os fractais apresentam, por definição, grau de detalhamento infinito.

3.6- Fractais e Imagens Digitais

A geometria fractal é muito utilizada em computação gráfica por fornecer uma base para a modelagem de objetos que apresentam grande variedade de detalhes, principalmente na representação de fenômenos naturais tais como: nuvens, montanhas, selvas, dentre outros. Ultimamente, as aplicações de técnicas baseadas em fractais têm sido propostas em várias

áreas do processamento digital de imagens (PDI), tais como segmentação, análise, síntese e codificação de imagens, esta última é a área de estudo deste trabalho.

Apesar de exigirem grande espaço de armazenamento e largura de faixa para transmissão, as imagens digitais possuem certas características muito exploradas no processo de codificação. Uma delas consiste na redundância relativa existente nos objetos que a compõem, ou seja, as imagens digitais possuem, na grande maioria das vezes, regiões auto-semelhantes, permitindo que tal propriedade possa ser explorada pela adoção de técnicas baseadas em fractais. Entretanto, a auto-similaridade presente nas imagens digitais difere da auto-similaridade discutida na geometria fractal. Hart [32] identificou algumas diferenças básicas entre a geometria fractal e a compressão fractal de imagens digitais.

A compressão de imagens por fractais foi inicialmente proposta por Michael Barnsley, posteriormente, Arnaud Jacquin propôs um esquema de compressão de imagens que utiliza codificação por fractais em blocos, que possibilita a compressão de qualquer imagem digital monocromática. Atualmente, diversas pesquisas sobre codificação fractal de imagens têm sido apresentadas e muitos pesquisadores acreditam ser uma técnica bastante promissora.

Nesta seção serão apresentados alguns conceitos matemáticos fundamentais, principalmente o teorema da colagem e a codificação IFS, que constituem a base teórica da codificação fractal de imagens digitais.

3.6.1- A Máquina Fotocopiadora

Um sistema de funções iterativas (IFS), cuja definição foi apresentada na seção 3.2.17 deste capítulo, produz um atrator de acordo com o código IFS estabelecido para o sistema.

Seja um IFS obtido por $\{IR^2; w_1, w_2, ..., w_N\}$ e seja $A_0 \subset IR^2$ um conjunto compacto não vazio, então pode-se obter sucessivos $A_n = W^{on}(A)$ da seguinte forma:

$$A_{n+1} = \bigcup_{j=1}^{N} w_j(A_n), \text{ para } n = 1, 2, 3, ... \tag{3.18}$$

A seqüência $A_n \subset H(IR^2)$ converge para o atrator do IFS.

Uma analogia bastante interessante para um melhor entendimento do princípio fractal foi realizada por Fisher [6], comparando um IFS com uma fotocopiadora especial. Imagine um tipo especial de máquina copiadora que reduza a imagem a ser copiada pela metade e reproduza-a três vezes sobre a cópia, como mostrado na Figura 3.8, adaptada de Fisher [6].

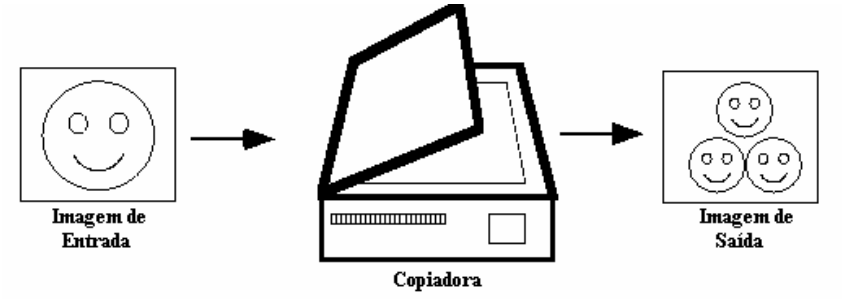


Figura 3.8- Copiadora que produz três cópias reduzidas da imagem de entrada.

Em seguida, utilize a imagem de saída como imagem de entrada, submetendo-a novamente à copiadora e assim por diante, sucessivas vezes. A Figura 3.9 ilustra este processo. A copiadora utilizada neste exemplo implementa o IFS utilizado para gerar o triângulo de Sierpinsk, como foi mostrado na seção 3.4.2 deste capítulo.

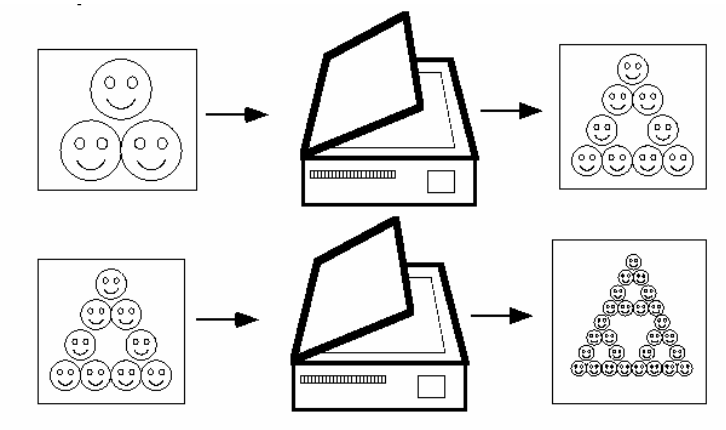


Figura 3.9- Processo recursivo de cópias.

Pode-se observar, na Figura 3.10, que qualquer subconjunto inicial compacto não vazio, de IR^2 converge essencialmente para um único ponto fixo, que é o atrator do IFS.

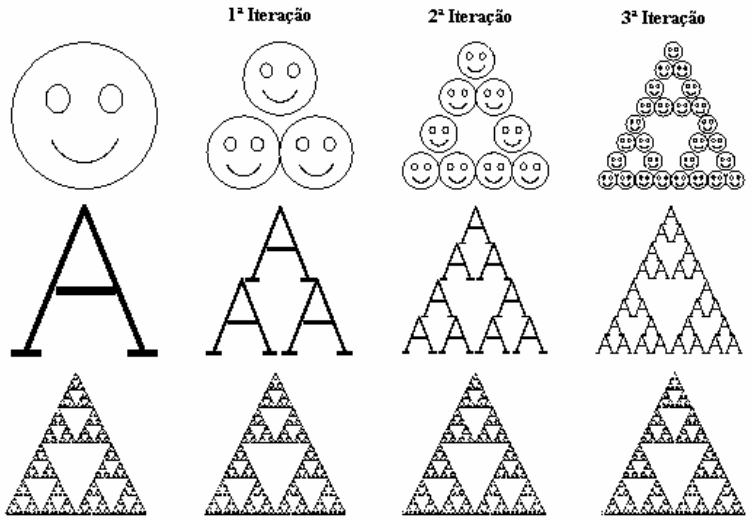


Figura 3.10- Primeiras iterações da máquina copiadora.

3.6.2- Teorema da Colagem

Na codificação de imagens digitais tem-se o chamado “problema inverso da teoria das transformações iterativas” [4, 5], a saber: seja *(X,d)* um espaço métrico completo de imagens digitais e considere uma imagem *T* a ser codificada, precisa-se obter uma contração *w* com fator de contração *s*, definida a partir de *(X, d)* para ele próprio ($w: X \rightarrow X$), para a qual *T* é um ponto fixo aproximado. Sendo assim, *w* precisa obedecer os seguintes requerimentos, estabelecidos na seção 3.2.13 deste capítulo:

$\exists\, 0 \leq s < 1 \mid \forall\, x,y \in X$ tem-se que $d(w(x),w(y)) \leq s.d(x,y)$ e

$d(T, w(T))$ seja tão pequeno quanto possível.

Sob essas condições, *w* é um código IFS para o atrator *T*, ou seja, *w* pode ser definido como um código de imagens com perda para a imagem original *T*.

O princípio da colagem é a base para a maioria dos sistemas de codificação fractal de imagens digitais, ele é fundamental para a obtenção de um IFS cujo atrator está próximo a um determinado subconjunto. Ou seja, o teorema da colagem possibilita encontrar um conjunto de transformações (contrações), com seus respectivos fatores de contração, cujo ponto fixo seja o atrator (fractal) [3].

Teorema da colagem: Seja *(X,d)* um espaço métrico completo e considere $T \in H(X)$ e $\varepsilon \geq 0$. Escolha um IFS $\{X; w_0, w_1, w_2, ..., w_N\}$ com fator de contração $0 \leq s \leq 1$, de modo que:

$h\left(T, \bigcup_{n=1_{(n=0)}}^{N} w_n(T)\right) \leq \varepsilon$, onde *h(d)* é a métrica de Hausdorff. Então

$$h(T, A) \leq \frac{\varepsilon}{1-s},$$

onde *A* é o atrator do IFS. Ou seja,

$$h(T, A) \leq (1 - s)^{-1} h\left(T, \bigcup_{n=1_{(n=0)}}^{N} w_n(T)\right) \tag{3.19}$$

para todo $T \in H(X)$.

O teorema da colagem estabelece que a distância entre um determinado $T \in H(X)$ e o atrator *A* obedece a Equação (3.19).

3.6.3- Codificação de Imagens pelo Princípio da Colagem (Codificação IFS)

O princípio da codificação fractal de imagens consiste na representação de uma imagem digital por um conjunto de transformações afins (contrações) cujo ponto fixo seja uma imagem bastante próxima da original [3]. Sendo o espaço de todas as imagens um espaço métrico completo com métrica de Hausdorff associada, o teorema do ponto fixo de Banach garante que um conjunto $W: H(X) \rightarrow H(X)$ de transformações contrativas possui um único ponto fixo *A*. Dessa forma, para qualquer imagem inicial $A_0 \in H(X)$, aplicando-se *W* sucessivamente em A_i $(i = 0, 1, 2, ..., n)$, converge-se à uma imagem fixa A_f. Ou seja

$$A_f = \lim_{n \to \infty} W^{on}(A_0). \tag{3.20}$$

A codificação IFS procura encontrar um código IFS para uma imagem *T* a ser codificada, que é gerado a partir de um conjunto de transformações afins (contrações) como representado pela Equação (3.21).

$$w_n(x) = w_n\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_n & b_n \\ c_n & d_n \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e_n \\ f_n \end{bmatrix}, n = 1,2,3.... \tag{3.21}$$

O algoritmo da codificação IFS inicia-se com a atribuiçao de valores para os coeficientes da transformação $w_1(x)$. A imagem $w_1(T)$ é obtida e analisada, caso $w_1(T)$ não se aproxime de uma parte de *T*, ajusta-se os coeficientes de $w_1(x)$ com a finalidade de se produzir uma nova $w_1(T)$ e assim sucessivamente, até que $w_1(T)$ aproxime-se de uma parte (região) de *T*. Quando isso acontece $w_1(T)$ está adequadamente posicionada e os coeficientes a_1, b_1, c_1, d_1, e_1 e f_1 são armazenados. A partir disso, uma nova transformação afim $w_2(x)$ pode ser ativada, produzindo um novo subconjunto $w_2(T)$. A imagem $w_2(T)$ é ajustada, pelos coeficientes de $w_2(x)$ até que a mesma seja um subconjunto de *T* não sobreposto a $w_1(T)$. Sobreposições entre $w_1(T)$ e $w_2(T)$ são permitidas, mas devem ser mínimas. Seguindo esse procedimento, existirão conjuntos $w_n(T)$ cuja união será uma imagem aproximada de *T*, que chama-se $\tilde{T}$, ou seja:

$$\tilde{T} = \bigcup_{n=1}^{N} w_n(T), \tag{3.22}$$

onde *N* é o número de contrações e deve ser o menor possível. A métrica utilizada para verificar a aproximação existente entre *T* e $\tilde{T}$ é a distância de Hausdorff $h(T, \tilde{T})$, que é inversamente proporcional ao grau de aproximação entre *T* e $\tilde{T}$. Os coeficientes das

transformações $\{w_1, w_2, ..., w_N\}$ são armazenados, sendo então o código IFS para a imagem T. O teorema da colagem assegura que o atrator A deste IFS será visualmente muito próximo de T. E ainda mais, se $T = \tilde{T}$, então $A = T$ [3].

A Figura 3.11 ilustra o princípio do teorema da colagem.

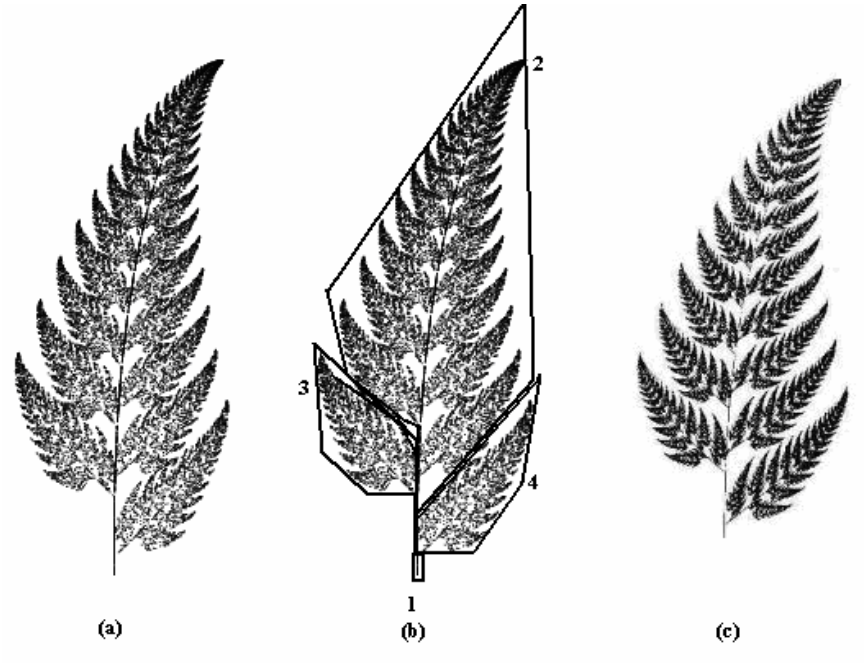


Figura 3.11 – (a) Imagem original; (b) colagem aproximada; (c) fractal associado.

Analisando a Figura 3.11(a), nota-se uma clara semelhança entre os frondes localizados de um lado e de outro da haste principal da folha de samambaia (regiões 3 e 4 da Figura 3.11-b). Verifica-se também, que ao omitir-se os primeiros frondes da esquerda e da direita obtem-se uma versão menor da folha de samambaia (região 2 da Figura 3.11 b). Além disso, percebe-se que os frondes de um lado e de outro são versões menores da folha de samambaia. A única região restante que não é similar a essas três regiões é a parte da haste principal destacada na região 1 da Figura 3.11(b). Aplicando-se o teorema da colagem obtém-

se o fractal ilustrado na Figura 3.11(c). Assim, o código IFS utilizado para gerar uma imagem aproximada da Figura 3.11(a) é composto pelos coeficientes de 4 transformações afins do tipo

$$w_n(x) = w_n\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} r\cos\theta & -s\,sen\,\theta \\ r\,sen\,\theta & s\cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} h \\ k \end{bmatrix}; n = 1,2,3,4.$$

Para exemplificar, um código IFS para a samambaia de Barnsley pode ser obtido pela Tabela 3.4 [3]:

Tabela 3.4- Código IFS para gerar a samambaia de Barnsley [3].

	Translação		Rotação		Escala	
w	h	k	θ	ϕ	r	s
1	*0.0*	*0.0*	*0*	*0*	*0.0*	*0.16*
2	*0.0*	*1.6*	*-2.5*	*-2.5*	*0.85*	*0.85*
3	*0.0*	*1.6*	*49*	*49*	*0.3*	*0.34*
4	*0.0*	*0.44*	*120*	*50*	*0.3*	*0.37*

A Figura 3.12 ilustra como a samambaia de Barnsley pode ser vista como o resultado de transformações afins de partes de si mesma.

A codificação IFS não se aplica na compressão de imagens reais, porque a auto-similaridade em imagens do mundo real não se apresenta de uma forma global, entretanto, é de fundamental importância para a compreensão da codificação do Sistema de Funções Iterativas Particionado (PIFS), utilizada nos esquemas de compressão fractal atuais.

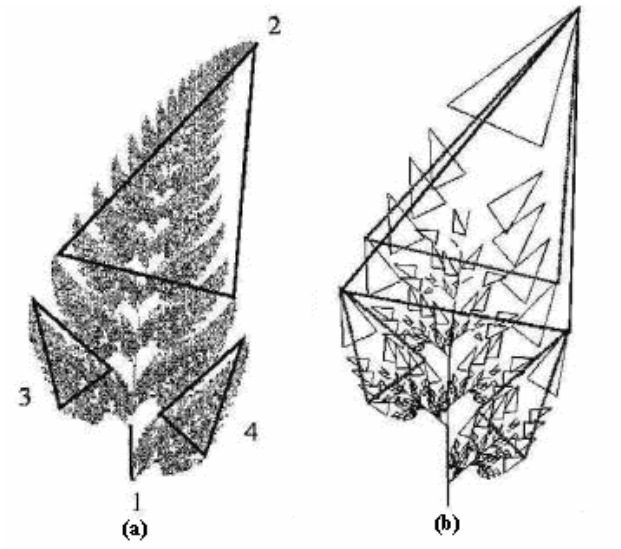


Figura 3.12- (a) Samambaia de Barnsley; (b) transformações afins.

3.6.4- Auto-similaridade em Imagens Digitais

As imagens digitais do mundo real, quase sempre, não possuem uma alto-semelhança como a definida na seção 3.5, naquele caso a imagem gerada apresenta uma auto-similaridade com os conjuntos menores dela mesma, portanto, ela é obtida a partir de cópias transformadas de si mesma, como pode ser visto nos exemplos apresentados na seção 3.4 deste capítulo. Entretanto, existe uma auto-semelhança entre algumas regiões das imagens, como ocorre na imagem mostrada Figura 3.13 [6].

Pode-se observar na Figura 3.13 (b) que as regiões destacadas apresentam uma auto-semelhança, possibilitando que a imagem seja representada por transformações afins de partes existentes nela mesma, ou seja, cópias de regiões transformadas de si mesma. Obviamente, as regiões auto-similares não são cópias perfeitas umas das outras, fazendo com que a imagem representada seja uma aproximação da original e não uma cópia fiel.

(a) (b)

Figura 3.13- (a) Imagem original; (b) regiões auto-similares.

Neste trabalho serão utilizadas imagens em escala de cinza com 8 bpp. Matematicamente, uma imagem digital é referida como uma função $z = f(x,y)$ que retorna o nível de cinza de cada píxel na posição (x,y). Usualmente, para imagens quadradas limita-se seu domínio ao quadrado unitário I^2, ou seja, $(x,y) \in \{(u,v): 0 \leq u,v \leq 1\} \equiv I^2$ e $f(x,y) \in I \equiv [0,1]$ [6].

3.6.5- Sistema de Funções Iterativas Particionado (IPFS)

A compressão fractal de imagens digitais tornou-se realidade na prática com a introdução do conceito de PIFS, apresentado por Jacquin [4, 5]. Enquanto no IFS proposto por Barnsley, baseado no teorema da colagem, o atrator é composto pela união de um conjunto de transformações afins (mapeamentos) sobre a imagem de entrada, no PIFS os mapeamentos operam sobre subconjuntos da imagem de entrada. Nesse sistema, a imagem é particionada em blocos chamados individualmente de bloco-imagem ou *range-block*, onde cada um é mapeado de um bloco-domínio ou *domain-block*. O conjunto de *domain-blocks*, chamado

domain pool, é considerado um *codebook* virtual, como pode ser visto na Figura 3.14 [27]. A combinação de mapeamentos consiste na transformação sobre a imagem como um todo.

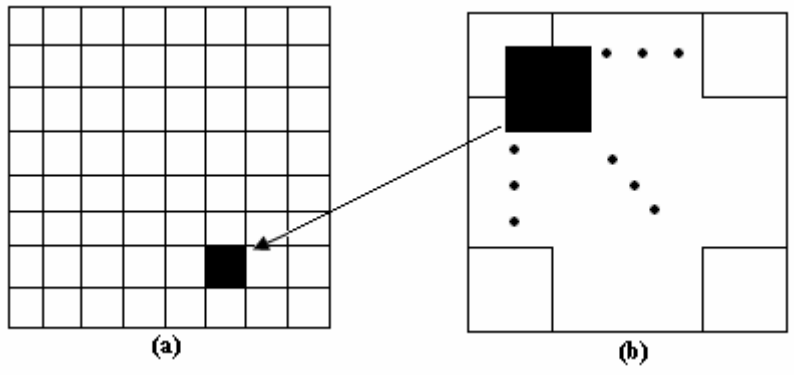


Figura 3.14- (a) *range-blocks* ; (b) *domain-blocks*.

Como o PIFS é a essência da codificação de imagens por fractais, o próximo capítulo será dedicado exclusivamente ao estudo desse sistema.

3.7- Considerações Finais deste Capítulo

A geometria fractal, com seus princípios estabelecidos, é uma área relativamente nova, entretanto, a necessidade de existir uma estrutura para descrever objetos com alto grau de irregularidades e complexidade já existe desde a segunda metade do século XIX.

Neste capítulo foram apresentados os conceitos e definições da geometria fractal, bem como alguns exemplos, suas principais características e fundamentação teórica sobre fractais relacionados a imagens digitais. Muitas pesquisas podem ser desenvolvidas sobre o assunto, visto que o estudo dos fractais propiciam uma alta gama de aplicabilidades. Neste trabalho, os fractais, bem como suas propriedades, serão explorados para a compressão de imagens

digitais, onde o conceito de sistema de funções iterativas (IFS), bem como sistema de funções iterativas particionado (PIFS), são os pontos chaves para o entendimento do processo.

O teorema da colagem é a base para a solução do problema inverso da teoria das transformações iterativas, por esse teorema tem-se o suporte matemático necessário para a compressão fractal de imagens. Entretanto, a auto-similaridade encontrada em uma imagem digital não se apresenta de forma global, sendo impossível obtê-la pela aplicação sucessiva de transformações afins, sobre cópias menores de si mesma, como acontece na geometria fractal. Não obstante, pode-se verificar que certas regiões das imagens são auto-semelhantes, podendo ser exploradas na codificação. Dessa forma, extendendo-se os conceitos de IFS para adaptar-se a essa propriedade, desenvolveu-se o PIFS, tornando-se possível a construção de um codificador fractal de imagens digitais.

O próximo capítulo apresenta os passos da codificação de imagens utilizando-se o PIFS e mostra os resultados obtidos ao utilizar essa técnica.

CAPÍTULO IV

CODIFICAÇÃO FRACTAL DE IMAGENS DIGITAIS

4.1- Introdução

O Sistema de Funções Iterativas Particionado (PIFS) juntamente com o teorema da colagem é a base da maioria dos algoritmos de codificação fractal de imagens digitais, a partir disso, várias pesquisas têm sido apresentadas e muitos estudiosos acreditam ser uma técnica bastante promissora, fazendo com que ela tenha se tornado uma das mais populares atualmente. O desenvolvimento de um codificador fractal de imagens envolve vários processos desde a escolha do tipo de partição para os *range-blocks* até o método de busca utilizado para associar cada *range-block* a um *domain-block*.

A primeira decisão a ser tomada para o desenvolvimento de um esquema de codificação fractal é a escolha do tipo de partição utilizado para os *range-blocks* e o *domain-pool* correspondente. A partir disso, divide-se a imagem em regiões (*domain-blocks*) e para cada região efetua-se as transformações geométricas e transformações de massa associadas, obtendo-se um *codebook* contendo as regiões transformadas de cada *domain-block*. Basicamente, o processo de codificação é pela comparação de cada *range-block* com todos os elementos do *codebook* e para aquele que mais se aproximar assume-se as transformações associadas para representar aquela região específica (*range-block*) da imagem, ou seja, para cada *range-block* existirá um conjunto de parâmetros de tranformações para representá-lo. A

decodificação é realizada pela aplicação iterativa das transformações em cada bloco, resultando a imagem decodificada. Neste capítulo serão apresentados os passos da codificação fractal de imagens digitais, bem como a demonstração de resultados utilizando-se esta técnica. Por se tratar de um codificador tradicional baseado nos modelos propostos por Arnaud Jacquin [4] e Yuval Fisher [6], neste capítulo tal esquema será referenciado como modelo de codificação fractal Jacquin-Fisher.

4.2- Partição de Imagens

Muitos tipos de particionamentos de *range-blocks* têm sido propostos [27], a maioria deles utiliza blocos retangulares. Os esquemas de partição mais comuns para os *range-blocks* são ilustrados na Figura 4.1.

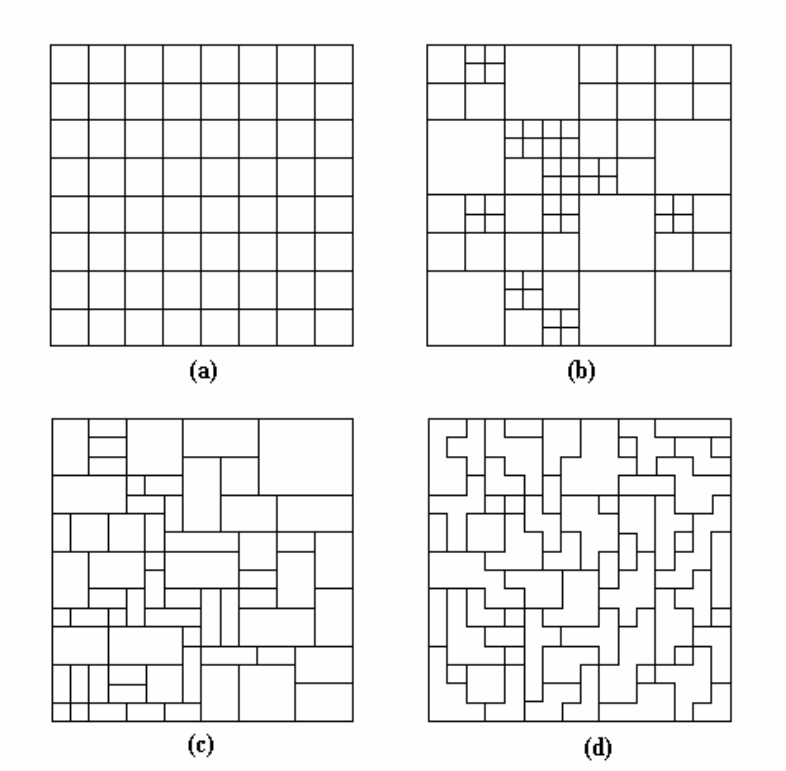


Figura 4.1- Tipos de partição de imagens: (a) blocos de tamanho fixo; (b) *quadtree*; (c) horizontal-vertical; (d) partição irregular.

A Figura 4.1(a) mostra um esquema de partição que utiliza blocos quadrados de tamanho fixo, utilizado em várias técnicas de codificação de imagens, como o JPEG, por exemplo. A Figura 4.1(b) apresenta o particionamento conhecido como *quadtree*, utilizado por Fisher [6], no qual os quadrantes de uma imagem são divididos por um processo recursivo, de forma que a imagem particionada pode ser representada por uma árvore em que cada nó não terminal possua quatro descendentes. No particionamento horizontal-vertical (HV), mostrado na Figura 4.1(c), a imagem também é particionada utilizando-se uma estrutura do tipo árvore, entretanto, isto não é realizado recursivamente, como acontece no *quadtree*. Na Figura 4.1(d) tem-se um esquema de partição que utiliza regiões irregulares que pode ser implementado pela combinação de outras estratégias, partindo-se inicialmente de blocos quadrados de tamanho fixo ou *quadtree* [27].

Existem esquemas de partições que utilizam blocos poligonais. Um grande número de partições baseadas em blocos triangulares tem sido estudado [27]. O processo inicia-se dividindo a imagem em dois triângulos, delimitados pela diagonal secundária, a partir disso, cada partição pode ser subdividida sucessivamente em triângulos menores por três divisórias de forma que um novo vértice é criado sobre cada um dos lados de um triângulo existente, como mostrado na Figura 4.2(a), ou por apenas uma divisória que parte de um dos vértices de um triângulo existente e termina sobre um novo vértice criado sobre o lado oposto do triângulo, como ilustra a Figura 4.2(b). Uma outra alternativa é baseada na triangulação de Delaunay, que define um conjunto inicial de “pontos semente” que é adaptado à imagem pela adição de pontos-semente nas regiões de alta variação, como exemplificado na Figura 4.2(c). A Figura 4.2(d) exibe uma partição poligonal construída pela subdivisão recursiva de uma grade inicial pela inserção de segmentos de reta em vários ângulos [27]. A triangulação de Delaunay é bastante utilizada como técnica de partição em muitos esquemas de compressão

de imagens [8, 33, 34], bastante indicada para aplicações de compressão de imagens por trabalhar de forma adaptativa, onde regiões da imagem com maior nível de detalhamento são divididas em um maior número de triângulos de tamanhos menores.

A utilização de *range-blocks* sobrepostos tem sido proposta com o intuito de reduzir o artefato conhecido como “blocagem” [27], muito freqüente nas imagens resultantes dos esquemas de codificação com perdas.

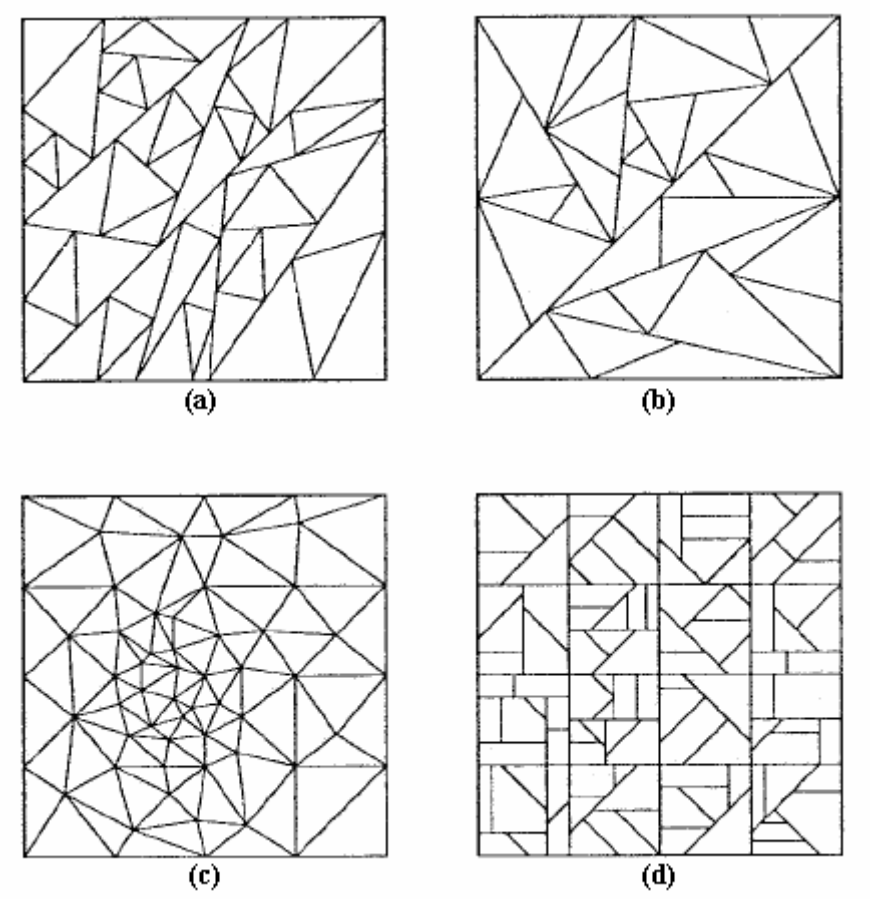


Figura 4.2- Tipos de partição de imagens: (a) Triangular (três divisórias); (b) Triangular (uma divisória); (c) Triangulação de Delaunay; (d) Poligonal.

Um exemplo simples de particionamento foi utilizado por Jacquin [4], ilustrado na Figura 4.3, onde são utilizados blocos quadrados não sobrepostos de tamanho fixo. Uma imagem original é dividida em células (blocos) de áreas quadradas não sobrepostas de duas formas, com 2 níveis: uma partição contendo blocos (*domain-blocks*) de tamanho *B* x *B* e outra contendo blocos (*range-blocks*) de tamanho *B/2* x *B/2*. Os tamanhos comumente utilizados são *domain-blocks* 16 x 16 e *range-blocks* 8 x 8.

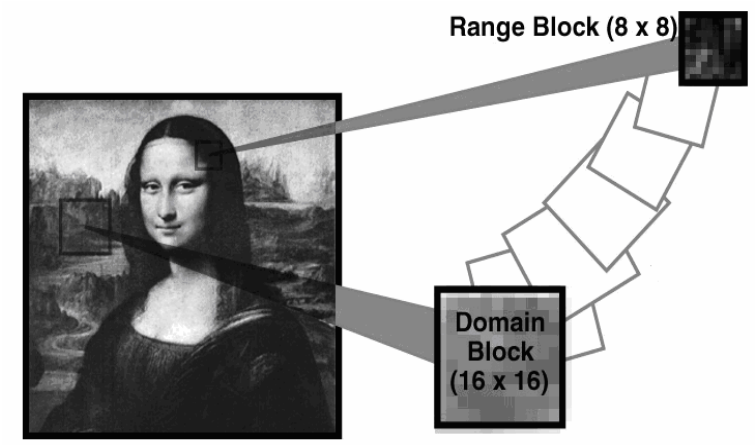


Figura 4.3- Esquema de partição em blocos de tamanho fixo com *range-blocks* 8x8 e *domain-blocks* 16x16.

A seleção de tamanhos e formas para os *range-blocks* depende de vários fatores. Para pequenos blocos de imagens, por exemplo 4 x 4, tem-se: 1) são fáceis de serem analisados e classificados geometricamente; 2) permitem uma rápida avaliação das distâncias entre blocos; 3) são fáceis de serem codificados com precisão; 4) conduzem a um sistema de codificação robusto, que mantém o desempenho mesmo para fontes de diversas imagens. Por outro lado, para grandes blocos tem-se: 1) permitem uma maior exploração da redundância em áreas de imagens suaves; 2) conduzem a maiores taxas de compressão [4].

4.2.1- Partição *Quadtree*

Existem muitos esquemas para o particionamento de imagens, desde o mais simples, como o particionamento por blocos quadrados de tamanho fixo, até os mais complexos, como as partições irregulares. A *quadtree* é um tipo de partição que fornece uma boa relação entre a taxa de compressão e a fidelidade entre a imagem original e a imagem reconstruída. Ela é a mais utilizada nas aplicações que envolvem a compressão fractal de imagens. O princípio básico consiste, inicialmente, em dividir a imagem original em quatro quadrantes iguais, posteriormente, cada quadrante pode ser dividido novamente em quatro subquadrantes e assim sucessivamente, até que um valor *rms* de tolerância ou um tamanho mínimo para cada subquadrante seja alcançado. Em outros termos, a imagem é representada por um tipo especial de árvore onde todos os nós ou são nós folha ou têm quatro nós filho, onde a imagem inicial é o nó pai, enquanto os quatro quadrantes são representados por quatro nós filho, em uma ordem pré-determinada. A Figura 4.4 ilustra uma idéia de como é a partição *quadtree*.

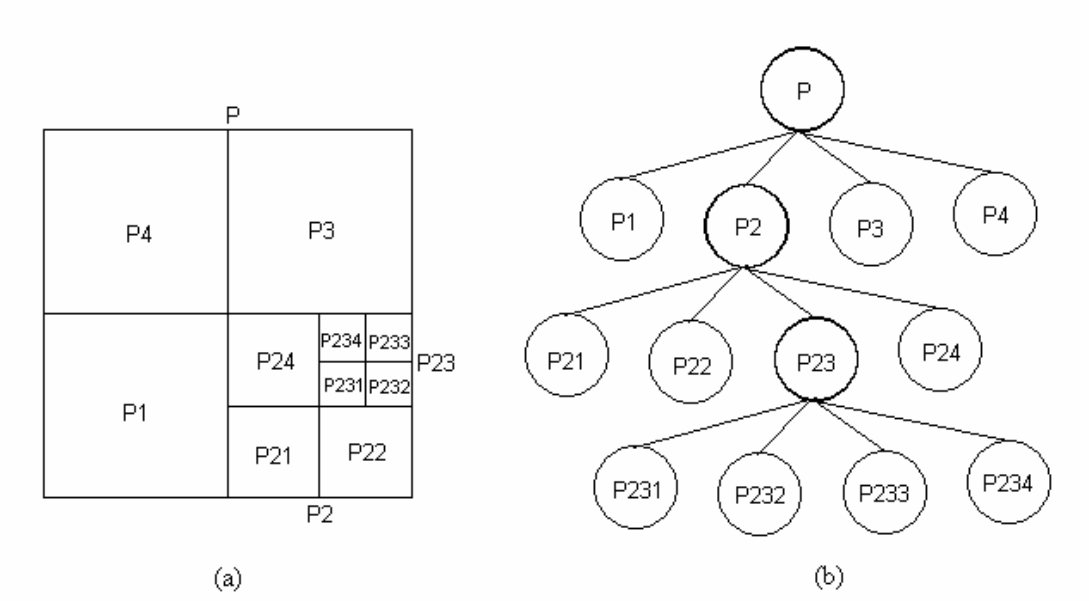


Figura 4.4- (a) Partição *quadtree* de três níveis; (b) árvore correspondente.

Pode-se observar, na Figura 4.4(a) que uma imagem P foi dividida em quatro quadrantes: P_1, P_2, P_3 e P_4, em seguida, o quadrante P_2 dividiu-se em quatro subquadrantes: P_{21}, P_{22}, P_{23} e P_{24}, finalmente, o quadrante P_{23} dividiu-se em outros quatro subquadrantes. A Figura 4.4(b) ilustra, em forma de árvore, esses três primeiros níveis. Para ilustrar um exemplo prático, na Figura 4.5, a imagem Lena foi particionada utilizando-se a *quadtree* [35].

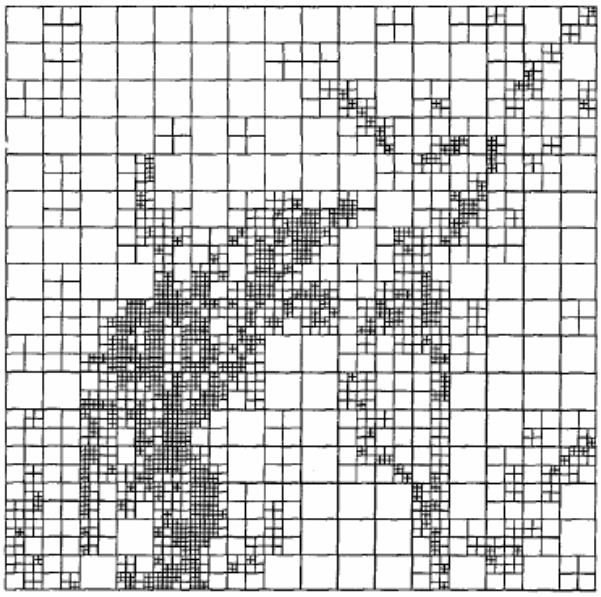


Figura 4.5- Exemplo prático do particionamento *quadtree*.

4.3- Transformações dos Blocos

A escolha do tipo de transformação é um ponto crítico no desenvolvimento de um codificador fractal em blocos. Aqui é necessária a distinção entre as transformações geométricas e as de massa, que operam diretamente sobre os pixels do *range-block*, alterando seus valores (massa ou luminância) [4, 5].

Suponha que uma imagem digital seja dividida em blocos quadrados não sobrepostos de tamanho *B x B* (*range-blocks*). Na codificação fractal, cada *range-block* R_i é codificado

segundo uma transformação contrativa F_i, composta por duas partes: parte geométrica T_i e parte de massa S_i, ou seja:

$$F_i = T_i + S_i \tag{4.1}$$

Seja $S(i_0,j_0,B)$ um *range-block* de tamanho B x B, com o extremo inferior esquerdo na posição (i_0, j_0), a parte geométrica S_i de F_i consiste na forma discreta de D, que é uma contração espacial de B pelo operador S, que mapeia os blocos de imagem de um *domain-block* $D_i = S(i_d,j_d,D)$ para um *range-block* $R_i = S(i_r,j_r,B)$. Quando $D = 2B$, os valores dos pixels da imagem contraída no *range-block* R_i são os valores médios dos quatro pixels no *domain-block* [4] :

$$(S_\mu)_{ir+i,\, jr+j} = (\mu_{I(i),J(j)} + \mu_{I(i)+1,J(j)} + \mu_{I(i),J(j)+1} + \mu_{I(i)+1,J(j)+1}) / 4\ , \tag{4.2}$$

para $i, j = 0, 1, ..., B-1$, onde as funções de índices I e J são obtidas por:

$$I(i) = i_d + 2i \qquad \text{e} \qquad J(j) = j_d + 2j \tag{4.3}$$

A parte de massa T_i de F_i são transformações sobre os *range-blocks*. Algumas transformações simples definidas no espaço dos *range-blocks* são definidas a seguir [4]:

- Absorção do nível de cinza $g_0 \in \{v_k\}_{0\le k\le G}$:

$$(\theta\mu)_{i,j} = g_0 \tag{4.4}$$

- Deslocamento da luminância pela constante $\Delta_g \in \{\pm v_k\}_{0\le k<G}$:

$$(\delta\mu)_{i,j} = \mu_{i,j} + \Delta_g \tag{4.5}$$

- Escalonamento do contraste por $\alpha \in [0,1]$:

$$(\sigma\mu)_{i,j} = \alpha\mu_{i,j} \tag{4.6}$$

As isometrias são transformações que alteram as posições dos pixels (embaralhamento) de modo determinístico em um *range-block*. A seguir serão apresentadas as oito isometrias canônicas de um bloco quadrado [4]:

1. Identidade

$$(\tau_0\mu)_{i,j} = \mu_{i,j} \tag{4.7}$$

2. Reflexão ortogonal em relação ao eixo médio vertical ($j = (B-1)/2$) do bloco:

$$(\tau_1\mu)_{i,j} = \mu_{i,(B-1-j)} \tag{4.8}$$

3. Reflexão ortogonal em relação ao eixo médio horizontal ($i = (B-1)/2$) do bloco:

$$(\tau_2\mu)_{i,j} = \mu_{(B-1-i),j} \tag{4.9}$$

4. Reflexão ortogonal em relação à diagonal principal ($i = j$) do bloco:

$$(\tau_3\mu)_{i,j} = \mu_{j,i} \tag{4.10}$$

5. Reflexão ortogonal em relação à diagonal secundária ($i + j = B-1$) do bloco:

$$(\tau_4\mu)_{i,j} = \mu_{(B-1-j),(B-1-i)} \tag{4.11}$$

6. Rotação de $+90^0$ em torno do centro do bloco:

$$(\tau_5\mu)_{i,j} = \mu_{j,(B-1-i)} \tag{4.12}$$

7. Rotação de $+180^0$ em torno do centro do bloco:

$$(\tau_6\mu)_{i,j} = \mu_{j,(B-1-i)} \tag{4.13}$$

8. Rotação de -90^0 em torno do centro do bloco:

$$(\tau_7\mu)_{i,j} = \mu_{(B-1-j),i} \tag{4.14}$$

Muitas outras isometrias mais complexas podem ser construídas, mas para um bloco de imagem bidimensional, basicamente são as 8 (oito) apresentadas. A Figura 4.6 ilustra as isometrias de uma imagem bidimensional.

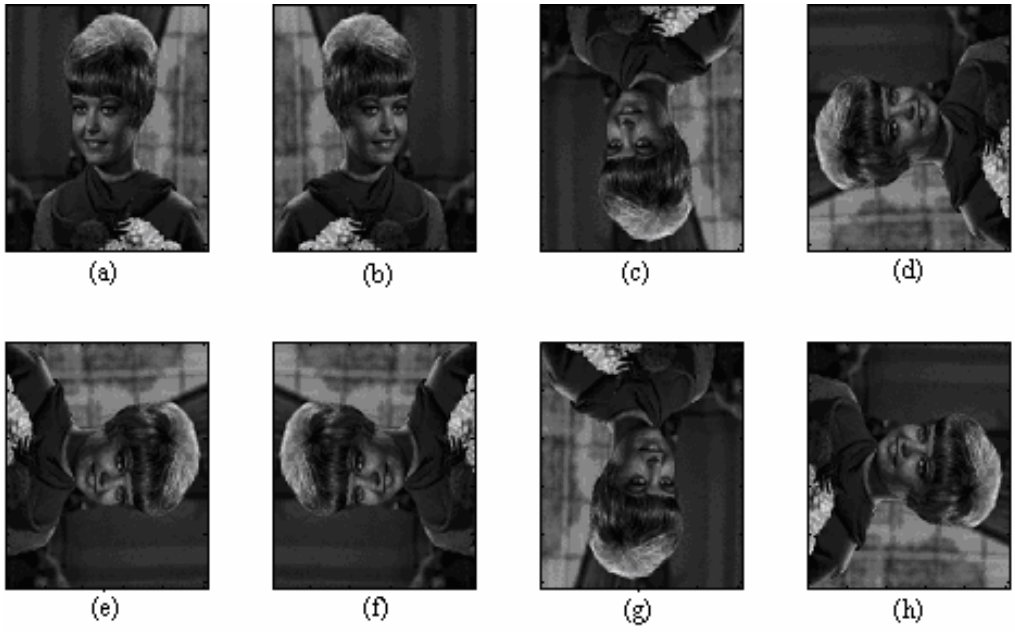


Figura 4.6- As oito isometrias; (a) identidade; (b) reflexão em relação à Y; (c) reflexão em relação à X; (d) reflexão em relação à diagonal principal; (e) reflexão em relação à diagonal secundária; (f) rotação de +90°; (g) rotação de +180°; (h) rotação de -90º.

Jacquin [4] enquadra as isometrias como sendo transformações de massa ou de intensidade, entretanto alguns pesquisadores classificam-nas como transformações geométricas ou transformações de bloco [27], por entenderem que elas atuam diretamente sobre a estrutura geométrica dos blocos da imagem.

4.4- Classificação dos Blocos

O grande esforço computacional necessário para a codificação fractal exige um tempo de processamento elevado e muitas pesquisas têm sido desenvolvidas visando apresentar técnicas para minimizar este problema. Grande parte do processamento é devido ao elevado tempo de busca gasto para associar um *range-block* a um *domain-block*, motivo pelo qual

várias estratégias para classificar os blocos do *domain-pool* têm sido apresentadas [36]. Durante a codificação, cada *range-block* também é classificado utilizando a mesma estratégia e só será comparado com os *domain-blocks* do mesmo tipo.

Jacquin [4] propôs um esquema de classificação, realizado por Ramamurthi e Gersho [37], baseado na intensidade dos níveis de cinza dos píxels dos *domain-blocks*, que foram definidos por três tipos de blocos: bloco de sombra (*shade*), com gradiente pouco significante; bloco de contorno (*edge*), com grandes variações de intensidade e; bloco intermediário (*midrange*), com gradiente moderado. Para cada *range-block R* na imagem original a ser codificado, primeiramente detecta-se a natureza de *R*, posteriormente cada *range-block* é tratado da seguinte forma:

Caso 1: *R* é um bloco de sombra – o *range-block R* é aproximado por um bloco de nível de cinza uniforme e igual ao nível médio de *R*, tal bloco é denotado por $\overline{R}$. A transformação δ_i que gera este bloco uniforme é simplesmente a absorção em $g_i=quant(R)$, onde a função "quant" representa a quantização uniforme na faixa de [0, 255].

Caso 2: *R* é um bloco intermediário ou um bloco de contorno – Cada bloco domínio *S(D)* do d*omain-pool* do mesmo tipo é analisado, a transformação é definida da seguinte forma:

Caso 2a : *R* é um bloco intemediário - a transformação de massa *T* é composta por um fator de escala de contraste e um deslocamento de luminância, da seguinte forma:

$$T_i(S(D_i)) = \alpha_i(S(D_i)) + \Delta g_i \tag{4.15}$$

onde α_i é o fator de escala de contraste e Δg_i é calculado tal que os níveis médios de cinza do bloco *R* e do bloco *D,* esse último multiplicado pelo fator de escala de contraste, sejam aproximadamente os mesmos, ou seja:

$$\Delta g_i = quant(\overline{R}_i - \alpha_i \overline{D}_i) \tag{4.16}$$

onde a função “quant” representa a quantização uniforme de 8 bits na faixa de [-128, 127].

Caso 2b: *R* é bloco de contorno – neste caso é realizada uma segmentação rude dos blocos *R* e *S(D)*, baseada no cálculo dos histogramas de níveis de cinza da imagem. Admite-se que blocos de imagem, de dimensões suficientemente pequenas, podem ser segmentados em duas regiões uniformes, uma escura e uma clara, separadas por uma região de transição. Os blocos segmentados são denotados por R_{seg} e $S(D_{seg})$. Após a segmentação, calcula-se a faixa dinâmica dos blocos segmentados, denotadas por $dr(R_{seg})$ e $dr(S(D_{seg}))$, como sendo a diferença dos níveis entre as regiões clara e escura. A transformação de massa *T* é composta por um fator de escala de contraste α, um deslocamento de luminância Δg e uma isometria τ, ou seja:

$$T_i(S(D_i)) = \tau_i(\alpha_i(S(D_i)) + \Delta g_i) \tag{4.17}$$

onde α_i é obtido por

$$\alpha_i = \frac{dr(R_{seg})}{dr(S(D_{seg}))} \tag{4.18}$$

Em seguida Δg_i é calculado e quantizado tal que os níveis de cinza das regiões escuras, ou das regiões claras, dependendo das populações dominantes de R_{seg} e $S(D_{seg})$, sejam os mesmos. Finalmente é selecionada uma dentre as oito isometrias. O bloco domínio D_i e a transformação de massa T_i que minimiza a distorção $d(R_i, T_i \circ S(D_i))$ são utilizados para representar o *range-block* R_i em questão.

Jacobs e Fisher [38] utilizaram uma classificação baseada na variância e média dos pixels. Primeiramente, cada bloco quadrado é dividido em quatro quadrantes: esquerdo alto, direito alto, esquerdo baixo e direito baixo, numerados seqüencialmente. Em cada quadrante, a média dos pixels A_i e a variância V_i correspondente são calculadas, da seguinte forma:

$$A_i = \frac{1}{n \times n} \sum_{j=1}^{n \times n} R_{i_j} \tag{4.19}$$

$$V_i = \frac{1}{n \times n} \left(\sum_{j=1}^{nxn} \left(R_{i_j} \right)^2 \right) - \left(A_i \right)^2 \tag{4.20}$$

onde R_{i_j} , $i = 1, 2, 3, 4$ e $j = 1, 2, 3, \ldots n$ x n , representa o *j-ésimo* pixel do *i-ésimo* quadrante. Ainda é possível orientar os blocos, pelas rotações e reflexões, em função dos valores da média de cada quadrante, sendo possível classificá-los em 3 (três) superclasses, ilustradas na Figura 4.7.

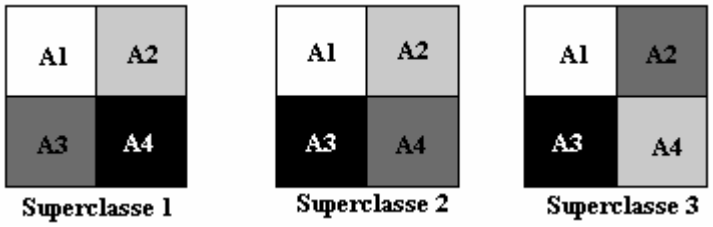


Figura 4.7 – Três tipos de superclasses.

De forma que:

Se $A1 \geq A2 \geq A3 \geq A4$ então Superclasse 1;

Se $A1 \geq A2 \geq A4 \geq A3$ então Superclasse 2;

Se $A1 \geq A3 \geq A2 \geq A4$ então Superclasse 3.

Além disso, cada Superclasse de média ainda apresenta 24 Subclasses de variância, que correspondem as 24 posições possíveis dos valores das variâncias distribuídos dentro do bloco, totalizando 72 subclasses, como ilustra a Figura 4.8. Este modelo de classificação é uma das principais características do codificador fractal de Fisher [7], muitas vezes referenciado como “acelerador de Fisher”.

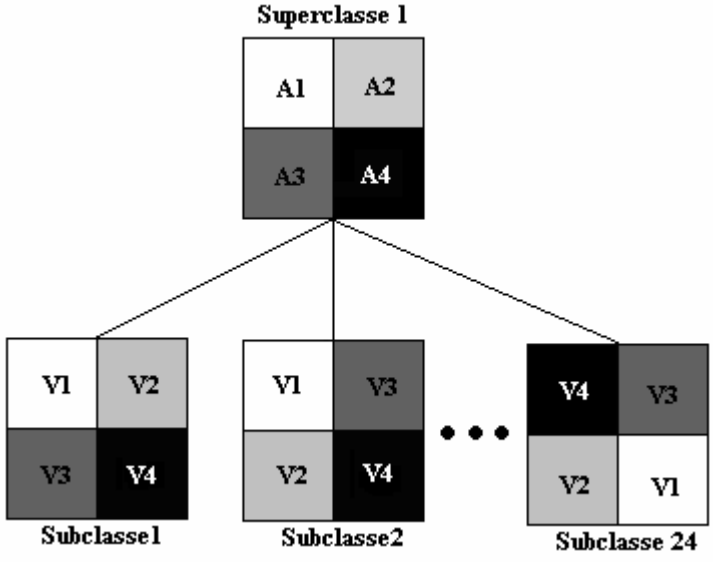


Figura 4.8 – Tipos de subclasses.

4.4.1- Classificação de Blocos Utilizada neste Trabalho

Neste trabalho é utilizado um outro tipo de classificação para os *domain-blocks* que também usa um método baseado em variância e média, proposto por Yung-Gi e outros [39], que é utilizado para classificar a simplicidade ou a complexidade de cada bloco. Em primeiro lugar os *domain-blocks* são classificados de acordo com a variância, pela Equação (4.20) e em seguida através da média, utilizando-se a Equação (4.19). Na codificação, cada *range-block R* é classificado conforme a diferença de variância $|Var(R)-Var(D')|$, onde *D'* representa o *domain-block* após a contração e possui o mesmo tamanho que *R*. Após isso, o *range-block é* classificado de acordo com o valor médio, fazendo com que os *domain-blocks* pesquisados para cada *range-block* limitem-se aos que possuam a mesma classe ou adjacentes, reduzindo o tempo de busca substancialmente. A Figura 4.9 ilustra esse processo.

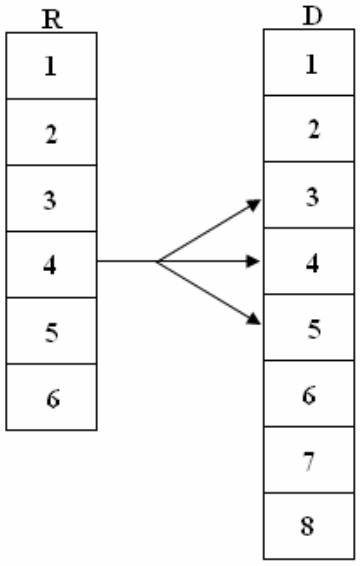


Figura 4.9 - Área de busca para a classe 4.

Muitas outras estratégias de classificação de blocos têm sido propostas visando minimizar o tempo de processamento utilizado na codificação fractal de imagens [27, 40, 41, 42, 43], e esse é o objetivo da grande maioria das pesquisas relacionadas nessa área.

4.5- Codificação Fractal de Imagens

O teorema da colagem, apresentado no item 3.6.2 do Capítulo 3 deste trabalho, é a essência da codificação fractal de imagens, que consiste em encontrar um mapeamento $w_1, w_2, \ldots, w_N$ com $W = \bigcup_{i=1}^{N} w_i$, cujo ponto fixo f' seja um conjunto muito próximo de uma imagem f a ser codificada, ou seja [6]:

$$f \approx f' \approx W(f') \approx W(f) \approx w_1(f) \cup w_2(f) \cup \cdots \cup w_N(f). \tag{4.21}$$

A representação matricial das transformações afins de um PIFS sobre uma imagem digital é da seguinte forma [6]:

$$w_i \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ o_i \end{bmatrix} \tag{4.22}$$

onde:

submatriz 2 x 2 $\begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix}$ - contração no plano x-y e isometria;

subvetor 2-D $\begin{bmatrix} e_i \\ f_i \end{bmatrix}$ - translação no plano x-y;

s_i - contraste;

o_i - brilho.

Então, é preciso obter uma imagem $f'=|W|$ cuja distância $\delta(f', f)$ seja mínima. Dessa forma, no PIFS é preciso encontrar um mapeamento W para cada *range-block* R_i da imagem a ser codificada, ou seja, encontrar o mínimo valor para a seguinte equação [6]:

$$\delta(f \cap (R_i \times I), w_i(f)) \quad i = 1,..., N . \tag{4.23}$$

A Equação (4.23) significa encontrar *domain-blocks* D_i e os respectivos mapeamentos w_i de forma que, quando aplicados à D_i, resultem em regiões muito próximas dos *range-blocks* R_i.

Na prática, um codificador fractal de imagens baseia-se na representação de uma região da imagem (*range-block*) pelas transformações afins de uma outra região (*domain-block*) da própria imagem, explorando ao máximo a propriedade de autosimilaridade presente nas imagens de uma forma geral. Uma aproximação para um *range-block* R_i a ser codificado pode ser expresso pela seguinte Equação (4.24) [9]:

$$R_i = s_i \cdot \tau_i\big(S(D_i)\big) + o_i \tag{4.24}$$

onde:

s: fator de contraste;

τ: isometria;

S: operador de decimação;

D: *domain-block*;

o: luminância.

O primeiro passo no processo de codificação fractal é particionar a imagem com o objetivo de designar o conjunto de *domain-blocks* (*domain-pool*) juntamente com as oito isometrias correspondentes de cada bloco D_i. Dessa forma, o *codebook* virtual (*domain-pool*) é composto pelos *domain-blocks*, que são blocos sobrepostos da própria imagem a ser codificada, mais as respectivas isometrias. Posteriormente, a imagem é dividida em blocos menores (*range-blocks*) não sobrepostos de forma que cada *range-block* é comparado com os elementos do *domain-pool*. Ou seja, é preciso encontrar o bloco D_i e a transformação $F_i = T_i + S_i$, que quando aplicada em D_i, resulta em um bloco muito próximo de R_i. Assim, cada *range-block* R_i é representado basicamente pelos seguintes parâmetros: posição do bloco domínio na imagem, a isometria associada, fator de contraste e a luminância.

Para escolher o *domain-block* D_i, juntamente com o fator de contraste s_i e a luminância o_i que melhor represente o *range-block* R_i é preciso encontrar um valor que minimize a distância *rms* entre eles, obtida pela Equação (4.25) [6]:

$$d_{rms} = \sum_{i=1}^{n} (s \cdot d_i + o - r_i)^2 \tag{4.25}$$

onde d e r correspondem ao *domain-block* e *range-block* transformados em vetores e $n=N \times N$ (dimensões do *range-block*). O menor d_{rms} ocorre quando as derivadas parciais em relação à s e o são iguais à zero, e para isso os cálculos de s e o são obtidos pelas Equações (4.26) e (4.27), respectivamente [6]:

$$s = \frac{\left[n^2 \left(\sum_{i=1}^{n} d_i \cdot r_i \right) - \left(\sum_{i=1}^{n} d_i \right) \cdot \left(\sum_{i=1}^{n} r_i \right) \right]}{\left[n^2 \sum_{i=1}^{n} d_i^{\,2} - \left(\sum_{i=1}^{n} d_i \right)^2 \right]} \tag{4.26}$$

e

$$o = \frac{\left[\sum_{i=1}^{n} r_i - s \cdot \sum_{i=1}^{n} d_i\right]}{n^2} \tag{4.27}$$

e neste caso d_{rms} é obtido pela Equação (4.28).

$$d_{rms} = \frac{\left[\sum_{i=1}^{n} r_i^2 + s \cdot \left(s \cdot \sum_{i=1}^{n} d_i^2 - 2 \cdot \left(\sum_{i=1}^{n} d_i \cdot r_i \right) + 2 \cdot o \sum_{i=1}^{n} d_i \right) + o \cdot \left(o \cdot n^2 - 2 \cdot \sum_{i=1}^{n} r_i \right)\right]}{n^2} \tag{4.28}$$

se $\left[n^2 \sum_{i=1}^{n} d_i^2 - \left(\sum_{i=1}^{n} d_i \right)^2 \right] = 0$, então $s = 0$ e $o = \frac{\sum_{i=1}^{n} r_i}{n^2}$ (4.29)

Para exemplificar o processo de codificação descrito acima, suponha uma imagem *f* de 512 x 512 pixels com 8 bpp (256 níveis de cinza). Seja *D* o conjunto de *domain-blocks* de 16 x 16 pixels cada, com sobreposição de 50% (8 x 8 pixels), ou seja, *D* é composto por um conjunto de $\left(\frac{512-(16-8)}{8}\right)$ x $\left(\frac{512-(16-8)}{8}\right)$ = 3969 blocos, então $D = D_1, D_2, ..., D_{3969}$, adicionando-se as 8 isometrias para cada D_i, resulta em um total de 8 * 3969 = 31752 elementos no *codebook* virtual. Seja *R* o conjunto de *range-blocks* (não sobrepostos) com 8x8 pixels cada um, ou seja, $R = R_1, R_2, ..., R_{4096}$, a codificação consiste em encontrar, para cada R_i, um $D_i \in D$ que minimize ao máximo a Equação (4.25), que na prática significa encontrar uma região D_i na imagem e um mapeamento w_i que mais se aproxima de R_i. Devido ao grande número de comparações efetuadas durante a codificação, atualmente muitas técnicas de

classificação de blocos têm sido apresentadas na literatura, conforme mencionado na seção 4.4 deste capítulo.

Neste exemplo os *domain-blocks* (16 x 16) são quatro vezes maiores do que os *range-blocks* (8 x 8), dessa forma, o operador S da Equação (4.24) representa uma contração espacial de D_i onde o valor de cada pixel é resultado da média de quatro pixels vizinhos, conforme apresentado na Equação (4.2). Com isso, para blocos de tamanho fixo, o funcionamento de um codificador fractal tradicional pode ser ilustrado na Figura 4.10.

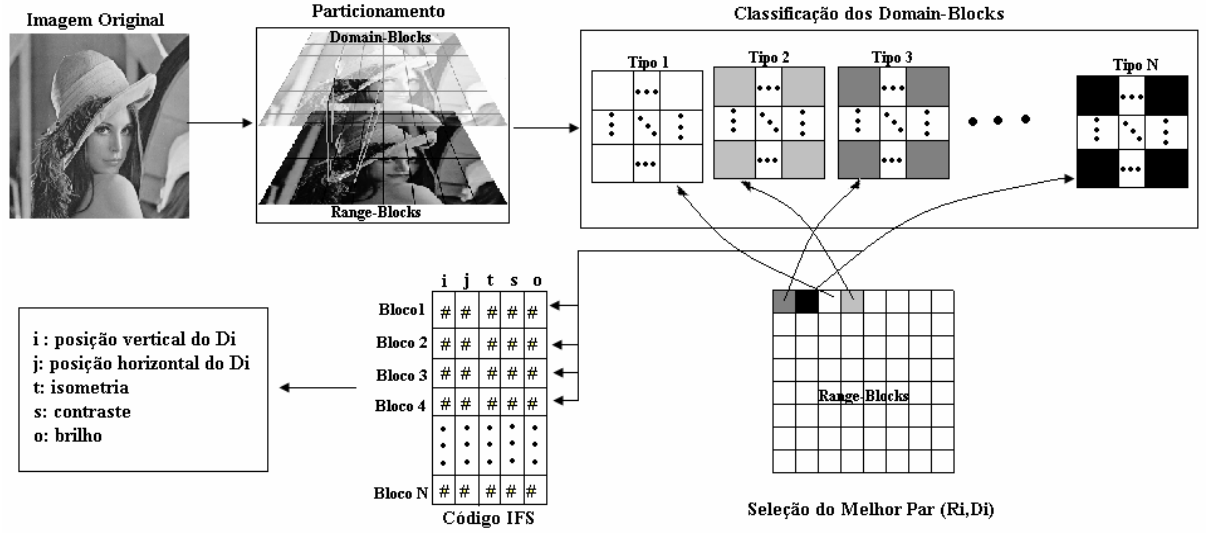


Figura 4.10 - Codificador fractal tradicional.

Para cada *range-block* utiliza-se 8 bits para armazenar a posição vertical (i) e 8 bits para a horizontal (j) do *domain-block* D_i respectivamente, 3 bits para a isometria τ_i, 7 bits para o brilho 0_i e 5 bits para o fator de contraste s_i , onde são necessário 31 bits para codificar cada *range-block* da imagem [6].

A decodificação consiste simplesmente em um processo iterativo do código fractal $W=\{w_1, w_2, ..., w_N\}$ (código IFS) sobre qualquer imagem inicial μ_0, até ser observada uma

convergência para uma imagem *f'* decodificada, ou seja, cada *range-block* decodificado R'_i é mapeado pela respectiva transformação w_i aplicada iterativamente à região da imagem μ_0 identificada pela posição do *domain-block* D_i durante a codificação. A seqüência de imagens $\{\mu_n = W^n(\mu_0)\}_{n=0}^{\infty}$ é chamada seqüência de reconstrução para o código *W*. A Figura 4.11 ilustra a decodificação da imagem Lena com 512 x 512 pixels de 8 bpp utilizando como imagem inicial um quadrado negro também de 512 x 512 pixels.

Figura 4.11- Seqüência de reconstrução da imagem Lena a partir de um quadrado negro.

Em geral, a convergência desejada é alcançada em torno de 10 a 15 iterações, partindo-se de qualquer imagem inicial. A Figura 4.12 ilustra novamente a decodificação da imagem Lena com 512 x 512 pixels, porém partindo-se de uma outra imagem inicial. Tais resultados demonstram na prática a fundamentação teórica apresentada nos Capítulos 3 e 4, constatando que um sistema de funções iterativas, quando aplicado a um conjunto inicial qualquer, tende a um mesmo ponto fixo, independente do conjunto inicial.

Figura 4.12- Seqüência de reconstrução da imagem Lena a partir de uma imagem qualquer.

4.6- Considerações finais deste Capítulo

Os trabalhos desenvolvidos por Jacquin e Fisher, introduziram o primeiro codificador fractal de imagens pela implementação prática do PIFS, esses trabalhos foram baseados no suporte teórico formulado por Barnsley e foi o passo inicial para uma série de pesquisas relacionadas a esta técnica emergente de compressão de imagens, a partir disso várias melhorias se fizeram presentes. Muitos modelos híbridos de codificação baseados em fractais têm sido apresentados e nota-se que a grande maioria visa a utilização de outras técnicas, em conjunto com a compressão fractal, para minimizar o esforço computacional necessário para a codificação, entre elas a quantização vetorial (VQ) [8, 10, 11], que é muito explorada por apresentar algumas características semelhantes à codificação fractal.

O próximo capítulo apresenta um estudo da codificação de imagens digitais utilizando-se a técnica de quantização vetorial e os resultados práticos obtidos pela implementação de um quantizador vetorial de imagens digitais.

CAPÍTULO V

QUANTIZAÇÃO VETORIAL (VQ) DE IMAGENS DIGITAIS

5.1- Introdução

A quantização é um dos últimos passos do processo de compressão com perdas. Nesta etapa, podem ser aplicados dois tipos diferentes: a quantização escalar ou a quantização vetorial. A quantização escalar baseia-se nas características estatísticas dos dados de entrada, onde cada elemento desse conjunto é tratado isoladamente [24]. Dessa forma, os dados são quantizados seletivamente, considerando que os elementos que carregam menos informação são quantizados grosseiramente ou até mesmo eliminados e os que apresentam maior importância são quantizados cuidadosamente.

Conforme a teoria de Shannon [44], em todos os aspectos, obtém-se um melhor desempenho codificando-se os vetores ao invés dos escalares. A quantização vetorial não considera cada elemento isoladamente, mas os agrupa em pequenos vetores (blocos) e, por um levantamento estatístico, determina quais vetores de dados são mais apropriados para formar a base de quantização.

A quantização escalar é bastante simples de se implementar e apresenta bom desempenho quando a largura de faixa utilizada é suficientemente grande comparada com os dados a serem transmitidos, entretanto, a quantização vetorial produz melhores taxas de compressão quando comparada com a quantização escalar.

Os quantizadores vetoriais podem ser classificados como sem memória ou com memória. Na VQ sem memória cada vetor é codificado de maneira independente, sem a influência dos vetores codificados anteriormente, bem como das saídas produzidas por eles. Já na VQ com memória existe um *feedback* após a codificação de cada vetor. Neste trabalho será utilizada a VQ sem memória, que pode ser vista como uma generalização do método I de Lloyd para quantizadores escalares ou mesmo uma extensão de um sistema *Pulse Code Modulation* (PCM) adaptado para o processamento vetorial. [13]

Neste capítulo, é apresentado um estudo da codificação de imagens digitais utilizando-se a técnica de quantização vetorial (VQ). Um algoritmo para o desenvolvimento de um quantizador vetorial ótimo para dados empíricos foi proposto por Linde, Buzo e Gray [14], constantemente referenciado como algoritmo LBG. Este algoritmo é abordado na Seção 5.4 deste capítulo.

5.2- Definição Geral

Quantização vetorial (VQ) é uma técnica de compressão em que as amostras do conjunto de dados a ser codificado são agrupadas em vetores *k*-dimensionais. Cada vetor *k*-dimensional é chamado de vetor de entrada (*input-vector*) para o quantizador. O principal componente envolvido na VQ é um *codebook* que contém um conjunto de vetores (*codevectors*) que serão utilizados para representar a informação a ser codificada com uma distorção tão mínima quanto possível. Dessa forma, para cada vetor de entrada X_i (*input-vector*) é selecionado o melhor vetor (*codevector*) do *codebook* que possa representá-lo. A partir disso, o índice *i* (*codeword*) do *codevector* é transmitido para o decodificador. Como o *codebook* está armazenado tanto no codificador como no decodificador, a decodificação é um

processo simples que envolve apenas a tarefa de buscar no *codebook* os vetores correspondentes aos índices transmitidos, obtendo-se assim a informação reconstruída (com perdas), representada pelo conjunto de vetores *Y*. A Figura 5.1 apresenta uma ilustração do funcionamento da VQ.

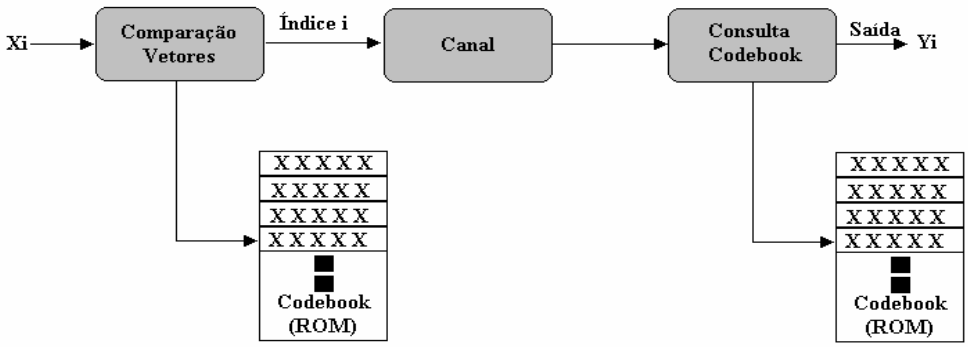


Figura 5.1- Diagrama de blocos de um quantizador vetorial simples.

Uma definição matemática para a VQ pode ser expressa da seguinte forma [13]: Um quantizador vetorial *k*-dimensional é a combinação de duas funções: um codificador γ que associa a cada vetor de entrada $x = (x_0, x_1, ..., x_{k-1})$ um símbolo $\gamma(x)$ presente em um conjunto de *M* símbolos, e um decodificador β que, por sua vez, associa a cada símbolo *v* em *M* um valor existente no alfabeto $\hat{A}$, onde, o conjunto de símbolos, por conveniência, é um espaço de vetores binários e, consequentemente, *M* é o conjunto de todos os 2^R vetores binários *R*-dimensional. O alfabeto $\hat{A}$ pode ou não ser idêntico ao espaço de vetores de entrada.

Se *M* possuir *m* elementos, então, calculando-se $R = \log_2^m$ obtém-se a taxa do quantizador em bits para cada vetor e, dividindo-se *R* por *k* (*R*/*k*) é obtida a taxa em bits por

símbolo, ou, quando a entrada for uma amostra de um sinal de voz ou imagem, por exemplo, obtém-se a taxa em bits por amostra.

Objetivamente, pode-se definir a quantização vetorial da seguinte forma [14]: um quantizador *k*-dimensional de *N* níveis é uma função, *q*, que associa a cada vetor de entrada , $x = (x_0, x_1, ..., x_{k-1})$, um vetor de saída especificado, $\hat{x} = q(x)$, extraído de um alfabeto de reprodução finito, $\hat{A} = \{y_i; i=1, 2, ..., N\}$. O quantizador *q* é descrito completamente pelo alfabeto (ou *codebook*) $\hat{A}$ juntamente com a partição, $S = \{S_i; i = 1, 2, ..., N\}$, do espaço vetorial de entrada em conjuntos $S_i = \{x: q(x) = y_i\}$ de vetores de entrada, que são mapeados pelo *i*-ésimo vetor de reprodução (ou *codeword*). Baseado neste conceito, tal técnica é chamada de quantização vetorial ou quantização em blocos.

A Figura 5.2 mostra simbolicamente o funcionamento de um quantizador vetorial na compressão de dados.

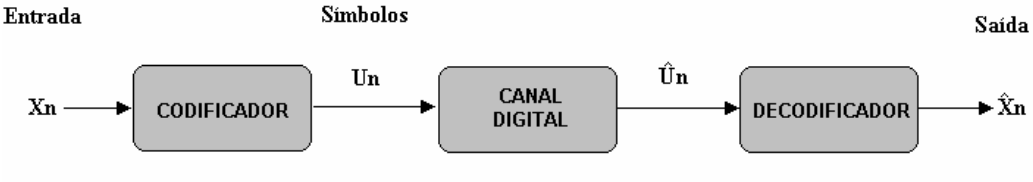


Figura 5.2– Sistema de compressão de dados utilizando-se quantização vetorial.

A informação de entrada $\{X_n; n= 0, 1, 2, ...\}$ é uma seqüência de vetores. O codificador produz uma seqüência de símbolos $\{U_n; n = 0, 1, 2, ...\}$. A seqüência $\{\hat{U}_n; n = 0, 1, 2, ...\}$ é transmitida ao receptor pelo canal digital. O decodificador então mapeia esta seqüência em uma seqüência de vetores de reprodução $\{\hat{X}_n; n = 0,1, 2, ...\}$ que formam a saída do sistema.

Os vetores de entrada podem ser amostras consecutivas de um sinal de voz ou de imagem, ou derivados de um sistema codificador. Neste trabalho, o quantizador possui como entrada vetores compostos pelos pixels de uma imagem digital.

O objetivo fundamental de um sistema de quantização é obter a seqüência de saída ($\hat{U}_n$) o mais próximo possível da seqüência original (U_n) para uma determinada taxa de compressão (R).

5.3- Distorção

O desempenho de um quantizador pode ser definido medindo-se a distorção $d(x, \hat{x})$ entre o vetor de entrada x e o vetor de saída $\hat{x}$ correspondente. Assume-se que a distorção causada, ao obter-se um vetor de saída $\hat{x}$ de um vetor de entrada x, é obtida por uma medida não negativa $d(x, \hat{x})$. Muitas medidas de distorção têm sido propostas [13]. A mais comum, por conveniência matemática, é a distorção erro-quadrático, obtida por:

$$d(x, \hat{x}) = \sum_{i=0}^{k-1} | x_i - \hat{x}_i |^2 \qquad (5.1)$$

Com a medida de distorção erro-quadrático pode-se quantificar o desempenho do sistema de quantização calculando-se a distorção esperada $\{E(d(X, \hat{X}))\}$ entre a entrada X e a saída reproduzida $q(X) = \hat{X}$. O sistema é bom se tiver uma pequena taxa de distorção. Seja $X = (X_0, X_1, ..., Xk_{-1})$ o vetor descrito por uma função de distribuição de probabilidade cumulativa $F(x) = Pr\{X_i \leq x_i ; i = 0,1, 2, ..., k-1\}$, a medida do desempenho de quantização q aplicada ao vetor X é obtida pela distorção esperada [6]:

$$D(q) = E(d(X, q(X)) = \frac{1}{N}\sum_{i=0}^{N-1} d(x_i, \hat{x}_i) \quad (5.2)$$

onde:

E - esperança referente à função de distribuição *F*.

5.4- O "Método I" de Lloyd para Quantizadores Escalares

Um quantizador q^* com *N* níveis é dito ótimo (ou globalmente ótimo) se ele reduzir ao mínimo possível a distorção esperada $D(q^*)$, isto é, q^* é ótimo se para todos os outros quantizadores com *N* níveis tem-se que $D(q^*) \leq D(q)$. Um quantizador *q* é dito localmente ótimo se *D(q)* for apenas um mínimo local, ou seja, pequenas mudanças em *q* provocam um aumento na distorção [14].

O propósito principal no desenvolvimento de um sistema de quantização vetorial é obter, se possível, um quantizador ótimo ou, caso contrário, obter um quantizador localmente ótimo e possivelmente bom [14]. Uma técnica para o desenvolvimento de um quantizador escalar proposta por Lloyd [45] é conhecida como "Método I". Em 1980, Linde, Buzo e Gray [14] estenderam a pesquisa de Lloyd para a quantização vetorial.

De forma geral, para descrever o *Método I*, assume-se que a distribuição é conhecida. Considere um quantizador *q* descrito pelo alfabeto de reprodução $\hat{A} = \{y_i ; i = 1,..., N\}$ e a partição $S = \{S_i; i = 1, ..., N\}$, então a distorção esperada $D(\{\hat{A}, S\}) \stackrel{\Delta}{=} D(q)$ pode ser escrita como [14]:

$$D(\{\hat{A}, S\}) = E(d(X, q(X)) = \sum_{i=1}^{N} E(d(X, y_i) \mid X \in S_i) \Pr(X \in S_i) \quad (5.3)$$

onde:

$E(d(X, y_i) \mid X \in S_i)$ - distorção esperada condicional, considerando que $X \in S_i$, ou equivalentemente $q(x) = y_i$.

Suponha que é considerado um alfabeto de reprodução particular $\hat{A}$, mas a partição S não é especificada. Uma partição S que é ótima para $\hat{A}$ pode ser construída facilmente a partir do mapeamento de cada x em $y_i \in \hat{A}$ minimizando-se a distorção $d(x, yi)$, isto é, pela escolha da distorção mínima ou da *codeword* mais próxima para cada entrada x. Se mais de uma *codeword* satisfizer tal condição, um método é utilizado para selecionar o índice mais baixo. A partição, chamada $P(\hat{A}) = \{P_i \; ; \; i = 1, ..., N\}$, construída desta forma é tal que $x \in P_i$ *(ou* $q(x) = y_i$*)* se, e somente se, $d(x,y_i) \leq d(x,y_j)$, para todo j, então [14]:

$$D(\{\hat{A}, P(\hat{A})\}) = E(\min_{y \in \hat{A}} d(X, y)) \quad (5.4)$$

isto implica que, para qualquer partição S

$$D(\{\hat{A}, S\}) \geq D(\{\hat{A}, P(\hat{A})\}) \quad (5.5)$$

então, para um alfabeto de reprodução fixo $\hat{A}$, a melhor partição possível correspondente é $P(\hat{A})$.

De maneira oposta, suponha uma determinada partição $S = \{S_i \; ; \; i = 1, ..., N\}$ que descreve um quantizador. Considere também que a distorção e a distribuição são tal que, para cada conjunto S, com probabilidade não zero, no espaço Euclidiano k-dimensional, existe vetor distorção mínimo $\hat{x}(S)$ para o qual [14]:

$$E(d(X,\hat{x}(S)) \mid X \in S) = \min_{u} E(d(X,u) \mid X \in S). \tag{5.6}$$

onde $\hat{x}(S)$ é chamado de centróide ou centro de gravidade do conjunto *S*. Dessa forma, para uma partição fixa $S = \{S_i ; i = 1, ..., N\}$, nenhum alfabeto de reprodução $\hat{A} = \{y_i ; i = 1,..., N\}$ pode produzir uma média de distorção menor do que o alfabeto de reprodução $\hat{x}(S) \overset{\Delta}{=} \{\hat{x}(S_i); i = 1,..., N\}$ contendo os centróides da partição S, desde que [14]:

$$\begin{aligned} D(\{\hat{A}, S\}) &= \sum_{i=1}^{N} E(d(X, y_i) \mid X \in S_i) \Pr(X \in S_i) \\ &\geq \sum_{i=1}^{N} \min_{u} E(d(X,u) \mid X \in S_i) \Pr(X \in S_i) \\ &= D(\{\hat{x}(S), S\}) \end{aligned} \tag{5.7}$$

As Equações (5.5) e (5.7) sugerem um algoritmo natural para designar um bom quantizador (*codebook)* utilizando um processo iterativo, partindo-se de um quantizador inicial qualquer [14]. A seção 5.5 apresenta um algoritmo baseado nesse princípio.

5.5- Algoritmo Linde_Buzo_Gray (LBG)

O ponto principal da VQ consiste na construção de um bom *codebook* e existem muitas técnicas para esse objetivo [46], onde a principal delas é conhecida como algoritmo LBG, apresentada por Linde, Buzo e Gray em 1980, desde então ele é o um dos mais utilizados nas aplicações que utilizam quantização vetorial atualmente. Como mencionado na Seção 5.4, esse algoritmo é uma generalização do algoritmo de Lloyd e muitas vezes ele é referenciado como algoritmo *k*-médias (*k-means*).

O algoritmo LBG [14] para uma determinada seqüência de treinamento, com distribuição desconhecida, pode ser descrito da seguinte forma:

(1) Considere o número de níveis N e um limiar de distorção $\varepsilon \geq 0$. Assume-se um alfabeto de reprodução inicial $\hat{A}_0$ de N níveis, uma seqüência de treinamento (x_j; $j = 0, 1, \ldots, n-1$) , m o número de iterações, inicializado com valor igual a zero e a distorção D, inicializada com valor igual a -∞.

(2) Para um determinado $\hat{A}_m = (y_i;\ i = 1, \ldots, N)$ encontra-se a partição com distorção mínima $P(\hat{A}_m) = (S_i\ ;\ i = 1, \ldots, N)$ da sequência de treinamento: $x_j \in S_i$ se $d(x_j, y_i) \leq d(x_j, y_l)$, para todo l. Calcula-se a distorção média, que é obtida por:

$$D_m = D[(\hat{A}_m, P(\hat{A}_m))] = n - 1 \sum_{j=0}^{n-1} \min_{y \in A_m} d(x_j, y) \tag{5.8}$$

(3) Se $(D_{m-1} - D_m) / D_m \leq \varepsilon$, o processo é finalizado com $\hat{A}_m$ sendo o alfabeto de reprodução final. Caso contrário, o processo continua.

(4) O alfabeto de reprodução ótimo é encontrado por $\hat{x}(P(\hat{A}_m)) = (\hat{x}(S_i); i = 1, \ldots, N)$ para $P(\hat{A}_m)$ onde:

$$\hat{x}(S_i) = \frac{1}{\| S_i \|} \sum_{j: x_j \in S_i}^{m} x_j \tag{5.9}$$

(5) Efetua-se a atribuição: $\hat{A}_{m+1} = \hat{x}(S_i)$,incrementa-se m de 1 ($m = m + 1$); retorna-se ao passo 2.

Pode-se observar que o algoritmo LBG exige que um alfabeto de reprodução inicial $\hat{A}_0$ seja designado. Existem várias técnicas para obter-se o *codebook* inicial. A mais simples delas utiliza a própria seqüência de treinamento como *codebook* inicial. Linde [14] utilizou uma técnica de divisões sucessivas (*splitting*) em que o centróide para a seqüência de treinamento de entrada é calculado e então, ela é particionada em dois vetores. Os centróides das duas partições resultantes são calculados e cada partição é novamente dividida em duas. Tal procedimento é repetido até que um vetor de reprodução inicial com N níveis seja criado, como segue:

(1) Inicia-se fazendo $M = 1$ e define-se, $A_0(1) = \hat{x}(A)$, como sendo o centróide da seqüência de treinamento (alfabeto de entrada);

(2) Considere o alfabeto de reprodução, $\hat{A}_0(M)$, contendo M vetores $\{y_i = 1, ..., M\}$, particiona-se cada vetor y_i em dois vetores $y_i + \varepsilon$ e $y_i - \varepsilon$, onde ε é um vetor de perturbação fixo. A coleção $\tilde{A}$ de $\{ y_i + \varepsilon , y_i - \varepsilon ; i = 1, ..., M\}$ possui $2M$ vetores. Substitui-se M por $2M$.

(3) Compara-se M com N. Se $M = N$ então, $\hat{A}_0 = \tilde{A}(M)$ e $\hat{A}_0$ é o alfabeto de reprodução inicial para o algoritmo de quantização com N níveis. Caso contrário, executa-se o algoritmo para quantização de M níveis sobre $\tilde{A}(M)$, com a finalidade de produzir um bom alfabeto de reprodução $\hat{A}_0(M)$, e retorna-se ao passo (2).

Utilizando-se o algoritmo *splitting* em uma seqüência de treinamento qualquer, inicialmente obtém-se um quantizador de 1 (um) nível, que consiste do centróide da sequência de treinamento. Este vetor é dividido em dois vetores e o algoritmo para quantização de 2 níveis é executado sobre esse par de vetores para obter-se um quantizador de 2 níveis. Cada

um desses dois vetores então são particionados e o algoritmo é executado com a finalidade de produzir-se um bom quantizador de 4 níveis. O processo é repetido até que um quantizador de *N* níveis seja obtido.

5.6- Técnicas de Quantização

A complexidade da VQ é medida em função do algoritmo que implementa a codificação e a decodificação dos dados. Para comparar os algoritmos, algumas métricas são levadas em consideração, tais como [15]:

A - número de multiplicações por amostra (ou pixel);

M - número de bytes de memória requeridos para o armazenamento;

R - taxa de bits;

D - distorção.

Para uma quantização vetorial k-dimensional com taxa fixa, com um *codebook C* contendo 2^{kR} *codevectors*, utilizando-se o método da força bruta para a codificação, é necessário calcular a distorção entre cada vetor de entrada *x* com todos os vetores de *C*, onde o *codevector* $\hat{x}$ selecionado é aquele que minimiza a distância $d(x, \hat{x})$. Nesse tipo de VQ o armazenamento *M* e a complexidade aritmética *A* aumentam exponencialmente em função de *k* e *R*, dessa forma, muitas vezes é necessário implementar uma estrutura para o *codebook* com a finalidade de acelerar o processo de busca da melhor *codeword*, reduzindo a complexidade aritmética (*A*) e a memória necessária para o armazenamento (*M*). Diferentes técnicas de quantização surgiram nos últimos anos e são discutidas nas próximas subseções deste capítulo [15].

5.6.1- VQ Estruturada por Árvore (*Tree-Structured Vector Quantization - TSVQ*)

A TSVQ foi proposta primeiramente por Buzo, A. H. Gray, R. M. Gray e Markel [47]. Uma k-dimensional TSVQ é um quantizador de taxa fixa *R* onde a codificação é realizada baseada em um *codebook* cuja estrutura é uma árvore binária de profundidade *kR*, na qual cada um dos 2^{kR} nós corresponde a uma *codeword* e cada um dos 2^{kR-1} nós interiores da árvore correspondem a um vetor teste, como ilustrado na Figura 5.3. A codificação de um vetor de entrada x é realizada por uma busca na árvore binária, de forma que um dentre os dois vetores teste é escolhido em cada nível até encontrar um nó terminal, sendo esse o *codevector* $\hat{x}$ escolhido cujo índice é transmitido para o decodificador. A decodificação é realizada simplesmente consultando o *codebook* e recuperando os vetores selecionados.

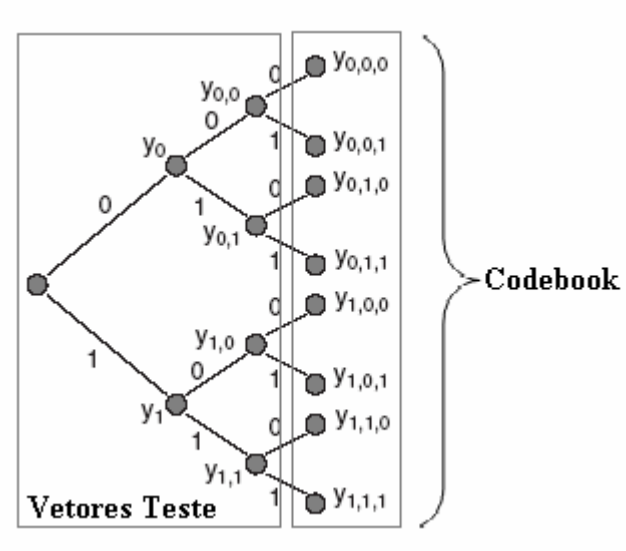


Figura 5.3– Àrvore da TSVQ.

A árvore binária da estrutura TSVQ precisa estar armazenada no codificador e requer aproximadamente o dobro do espaço requerido pela VQ sem estrutura, entretanto a complexidade aritmética é reduzida substancialmente. Contudo, as *codewords* escolhidas

muitas vezes podem não serem as melhores em termos de distorção, dependendo muito do processo de decisão utilizado na árvore binária. A fidelidade e a complexidade da TSVQ aumentam de acordo com a dimensão e a ordem da árvore.

5.6.2- VQ Estruturada por Produto (*Product Vector Quantization*)

A VQ estruturada por produto utiliza um *codebook* resultante do produto cartesiano de dois outros *codebooks* de dimensões menores. A complexidade é obtida pela soma das complexidades dos dois *codebooks* menores, que é inferior à complexidade de uma VQ sem estrutura, mantendo o mesmo número de *codevectors*. O problema desta técnica é a divisão da taxa de alocação de bits entre os dois *codebooks*. Uma alternativa é organizar o conjunto de *codevectors* como o produto cartesiano entre um *codebook* vetorial, que descreve a forma de cada *codevector* (*shape codebook*), e um *codebook* escalar, que descreve o ganho do *codevector* (*gain codebook*). Um exemplo de uma estrutura deste tipo, conhecida como *shape-gain* VQ (SGVQ) é ilustrado no diagrama de blocos da Figura 5.4 [15].

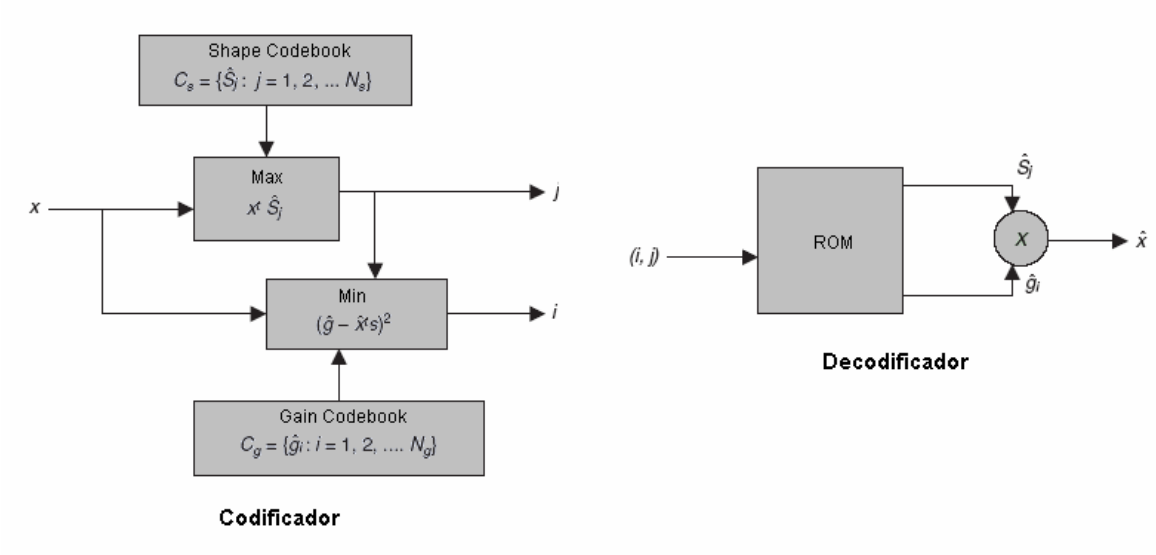


Figura 5.4– Codificador e decodificador SGVQ.

O codificador da SGVQ divide o vetor de entrada *x* em um par indexado (*i*,*j*) e o transmite, após a transmissão o decodificador busca em cada um dos *codebooks* (*shape* e *gain*) as *codewords* correspondentes e multiplica uma pela outra para obter o *codevector*.

A redução da complexidade computacional e de memória para armazenamento torna a SGVQ bastante atrativa, podendo ser utilizada em aplicações de codificação de voz [48].

5.6.3- VQ Estruturada por Endereço (*Address Vector Quantization - AVQ*)

A AVQ é um tipo de VQ com memória que é utilizada na codificação de imagens pela exploração da redundância estatística existente entre um grupo de blocos vizinhos, ou seja, explora a correlação existente entre os blocos vizinhos. Um *codebook* de endereços é utilizado para codificar um grupo de blocos, onde cada elemento desse *codebook* é um endereço para um vetor do *codebook* LBG. O *codebook* de endereços é composto por duas regiões: uma endereçável chamada região ativa e outra não endereçável (região inativa), como ilustrado na Figura 5.5.

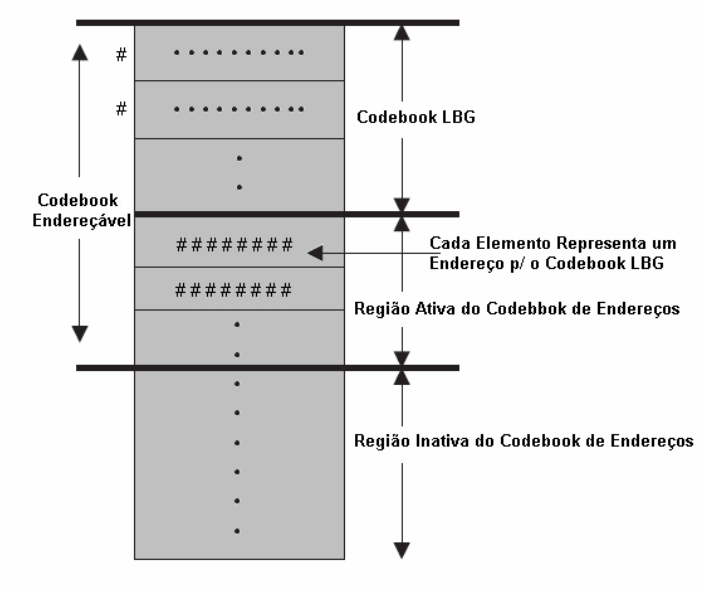


Figura 5.5– *Codebook* da AVQ.

Durante a codificação, todas as entradas da região ativa do *codebook* de endereços são verificadas e, caso exista algum vetor selecionado, o índice do *codebook* de endereços é transmitido, senão, o endereço de cada bloco é transmitido. A partir disso, os *codevectors* do *codebook* de endereços são reorganizados adaptativamente com a finalidade de trazer o vetor de endereço mais provável para a região ativa. O decodificador contém o *codebook* de endereços e o *codebook* LBG e a decodificação é realizada consultando-os e extraindo os vetores selecionados, como é realizado na VQ tradicional.

A taxa de bits da AVQ é aproximadamente a metade da VQ simples mantendo a mesma qualidade da imagem codificada e o custo computacional para encontrar os endereços no *codebook* de endereços envolve uma comparação binária simples, entretanto a reordenação do *codebook* de endereços durante a codificação aumenta a complexidade computacional, que pode ser minimizada reduzindo-se seu tamanho. Uma outra desvantagem é devido a necessidade de sincronia entre os *codebooks* de endereços do codificador e do decodificador. A AVQ pode ser vista com bastante detalhes em Narasbadi e Feng [49].

5.6.4- VQ Estruturada por Células (*Lattice Vector Quantization - LVQ*)

A LVQ pode ser considerada uma extensão da quantização escalar uniforme para vetores, onde ao invés de um elemento de cada vez como saída do quantizador, como é realizado na quantização escalar, uma sequência de vetores (subconjuntos de pontos) é a saída do quantizador vetorial. Um exemplo de uma estrutura de células é apresentado na Figura 5.6. Os subconjuntos precisam possuir formas regulares, onde é viável evitar lacunas entre as células, dessa forma, regiões hexagonais são as mais indicadas.

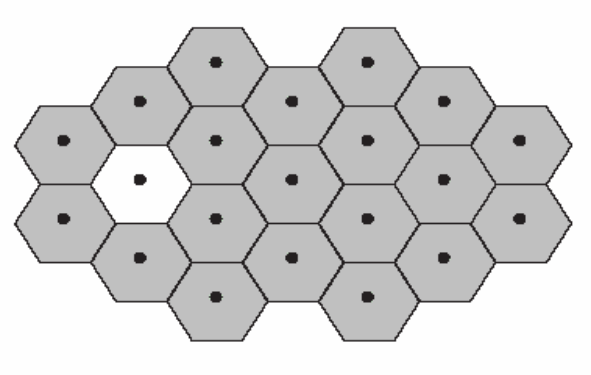


Figura 5.6– Estrutura em células da LVQ.

A estrutura regular da LVQ contribui para diminuir substancialmente o tempo gasto na comparação de um vetor de entrada com os vetores do *codebook*, isto acelera o processo de codificação. A fidelidade e a complexidade da LVQ aumentam de acordo com a dimensão e a forma escolhida para as células. A LVQ pode ser vista detalhadamente em Conway e Sloane [50, 51, 52].

5.6.5- VQ Preditiva (*Predictive Vector Quantization*)

A VQ preditiva é uma técnica de quantização vetorial com memória em que os *codebooks* múltiplos são armazenados, de forma que as saídas anteriores são utilizadas para determinar qual *codebook* será consultado para codificar a entrada atual. O processo de codificação consiste de duas operações, como mostrado na Figura 5.7:

1) Obter u_n resultante de uma entrada x_n no estado presente S_n;

2) Encontrado u_n, determinar o próximo estado S_{n+1}.

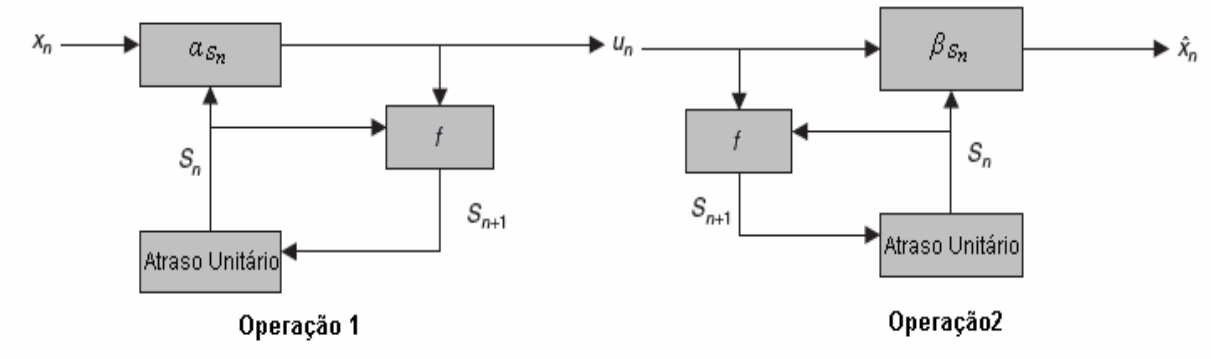


Figura 5.7– As duas operações da VQ preditiva.

A primeira operação VQ preditiva é exatamente a mesma de um quantizador vetorial *k*-dimensional sem memória, enquanto que a segunda é um mapeamento do conjunto de saídas em um conjunto de estados. A VQ preditiva requer o mesmo esforço computacional que a VQ sem memória, mas exige mais espaço de memória devido à necessidade de armazenamento de vários *codebooks* e da tabela de transição de estados, entretanto, proporciona melhores resultados que a VQ sem memória. O codificador e o decodificador precisam compartilhar um conjunto finito de estados e um quantizador designado para cada estado, de forma que o decodificador precisa saber qual codebook está sendo utilizado. A VQ preditiva pode ser vista mais detalhadamente em Haoui e Messerschmitt [53].

5.6.6- *Fine-Coarse VQ* (FCVQ)

A FCVQ é uma técnica de quantização vetorial que utiliza dois quantizadores, um quantizador mais refinado (quantizador fino ou *fine quantizer*) e um outro quantizador mais grosseiro (quantizador grosseiro ou *coarse quantizer*). O vetor de entrada *x* é primeiramente quantizado usando uma *fine* VQ, que pode ser uma quantização estruturada de qualquer tipo,

cujo *codebook* C_f é composto pelo conjunto de vetores $y_f=\{y_{f1}, y_{f2}, ..., y_{fNf}\}$. O segundo estágio consiste de uma *coarse* VQ com *codebook* $C=\{y_1, y_2, ..., y_N\}$, onde $N_f >> N$. O índice u_f correspondente ao *codevector* y_{fi}, resultante do primeiro estágio da quantização, serve de entrada para o segundo estágio e o índice *u*, correspondente ao *codevector* y_i é a saída do quantizador, como mostrado na Figura 5.8.

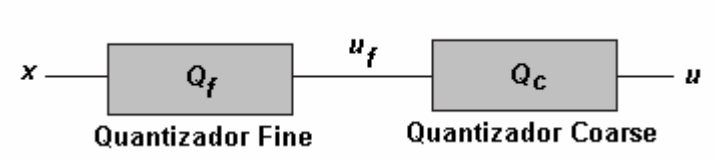


Figura 5.8– *Fine-coarse* VQ.

Quanto mais refinado é o primeiro estágio da quantização do FCVQ, melhor é o desempenho dele, contudo, ocorre também um aumento do espaço de memória necessário para o armazenamento. A complexidade aritmética é reduzida, comparando-se com outras técnicas estruturadas. Uma abordagem mais aprofundada sobre a FCVQ pode ser encontrada em Moayeri, Neuhoff e Stark [54].

5.6.7- VQ Estruturada por Múltiplos Estágios (*Multistage Vector Quantization – MSVQ*)

Um quantizador de múltiplos estágios é uma expansão da VQ original que minimiza a distorção com o ônus de um aumento da taxa de bits. Na MSVQ uma aproximação grosseira do vetor de entrada é produzida nos primeiros estágios, posteriormente, nos estágios seguintes, são realizadas sucessivas quantizações mais detalhadas. Um vetor de entrada *k*-dimensional é codificado no primeiro estágio e o erro entre o vetor original e o vetor quantizado é codificado no segundo estágio. Este processo pode ser repetido várias vezes com

estágios adicionais com a finalidade de maximizar a fidelidade entre o vetor transmitido e o vetor original [15]. Um exemplo de uma MSVQ de dois estágios é ilustrado na Figura 5.9.

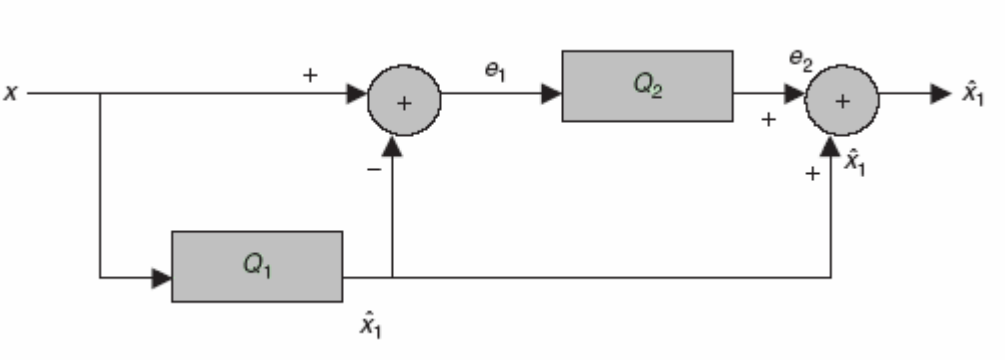


Figura 5.9– Quantizador vetorial de dois estágios.

O vetor de reprodução é obtido pela soma dos vetores de reprodução dos estágios anteriores e do novo resíduo quantizado. A taxa de bits da MSVQ é a soma das taxas dos estágios individuais, que causa o problema da alocação de bits entre os vários estágios. A distorção é obtida avaliando-se o vetor resultante do último estágio.

5.6.8- VQ Estruturada por Treliça (*Trellis-coded Vector Quantization – TCVQ*)

A VQ estruturada por treliça [55, 56] é caracterizada pelo aumento do desempenho e pela redução na complexidade de codificação. A TCVQ utiliza uma estrutura do tipo treliça, como mostra a Figura 5.10, que consiste de uma treliça de oito estágios composta por duas entradas e duas saídas para cada nó. Nesta estrutura, um *codebook* inicial é selecionado e posteriormente particionado em *subcodebooks* de forma que cada aresta da treliça corresponde a um *subcodebook*. O melhor vetor de reprodução é selecionado para uma

seqüência de vetores de entrada traçando-se o caminho que possui distorção mínima utilizando-se o algoritmo de Viterbi.

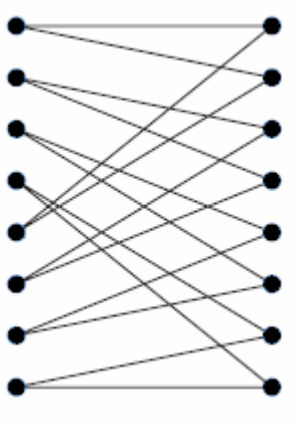


Figura 5.10– Treliça de oito estágios.

A TCVQ possui um *codebook* de tamanho 2^{kR+1}, mas em qualquer ponto um subconjunto de 2^{kR} vetores é utilizado como *codebook*, reduzindo-se assim a complexidade aritmética durante a codificação. O algoritmo LBG é uma boa alternativa para utilizar na TCVQ. O decodificador é uma máquina de estados cujo comportamento é direcionado pelas seguintes equações:

$$S_{n+1} = g(S_n, u_n) \tag{5.10}$$

$$\hat{x}_n = f(S_n, u_n) \tag{5.11}$$

onde n é um número inteiro que representa o índice de tempo, S_n e S_{n+1} representam o estado atual e o próximo estado respectivamente, u_n é a saída binária do codificador e $\hat{x}_n$ é o vetor de

reprodução. Maiores detalhes sobre a TCVQ podem ser vistos em Wang e Moayeri [55] e Khan, Smith e McLaughlin [56].

5.6.9- VQ Estruturada por Pirâmide (*Pyramid Vector Quantization - PVQ*)

A PVQ é um tipo de LVQ (*Lattice Vector Quantization*) que não é baseada em uma função de densidade de probabilidade uniforme, que pode ser construída baseada em um subconjunto de pontos obtidos pela fórmula de uma forma cúbica *S(k,L)*, obtida por:

$$S(k,L) = \left\{ x : \sum_{i=1}^{k} | x_i | = L \right\}, \tag{5.12}$$

onde *k* é a dimensão do vetor e *L* é um número inteiro positivo. Uma pirâmide com coordenadas inteiras sobre *S(3,4)* é mostrada na Figura 5.11.

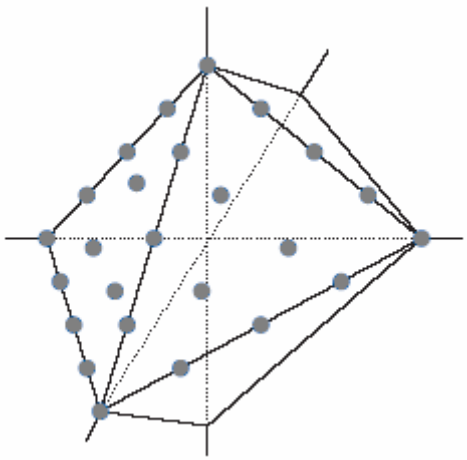


Figura 5.11– Pirâmide da PVQ.

O número de pontos da estrutura precisa ser avaliado e as *codewords* da PVQ são versões escalares das coordenadas que, para poderem ser utilizadas em comunicação digital, precisam ser representadas unicamente por seqüências binárias. Diversos algoritmos para este propósito estão disponíveis na literatura. A PVQ é apresentada com detalhes em Fisher [57].

Uma gama de técnicas de quantização está disponível na literatura e muitas delas podem ser combinações de duas ou mais técnicas discutidas anteriormente nesta seção. Uma comparação detalhada entre as técnicas descritas aqui pode ser consultada em Vasuki e Vanathi [15].

5.7- Quantização Vetorial de Imagens Digitais

O desempenho da quantização vetorial para imagens digitais tem sido amplamente estudado. O conceito básico consiste em particionar a imagem em vetores bidimensionais e cada vetor é comparado com um conjunto de vetores padrão (*codebook*) armazenados em ROM, então uma *codeword* que representa o vetor padrão que mais se aproxima do bloco da imagem é transmitido. O receptor, que também possui o *codebook* armazenado em ROM, reconstrói a imagem utilizando os vetores padrões ao invés dos vetores originais, como ilustrado na Figura 5.1. Cada vetor de entrada X_i é codificado por um *codebook*, que pode ser designado pelo algoritmo LBG, utilizando-se um critério de distorção.

A Figura 5.12 mostra as imagens utilizadas como base de treinamento para produzir um *codebook* de 256 níveis, utilizando-se o algoritmo LBG, o método *splitting* para gerar o *codebook* inicial e um limiar de distorção de 0,05.

Figura 5.12 – Base de treinamento para geração do *codebook.*

As imagens da Figura 5.13 apresentam tamanho 512 x 512 pixels, todas com 256 níveis de cinza (8 bits por pixel). A Figura 5.13 mostra os resultados obtidos, codificando-se a imagem Lena 512 x 512 pixels, com 256 níveis de cinza (8 bpp).

Figura 5.13– Resultados obtidos da VQ a 0,16 bpp.

A imagem original foi dividida em blocos (vetores) 8 x 8 que foram codificados mediante a comparação com o *codebook* gerado a partir da base de treinamento ilustrada na Figura 5.12. A imagem reconstruída a 0,16 bpp, apresenta PSNR de 27,64 dB em relação à imagem original com taxa de compressão de 50:1. A Figura 5.14 ilustra os resultados da mesma imagem Lena, porém utilizando-se blocos de tamanho 4 x 4. Pode-se verificar que a imagem reconstruída da Figura 5.12 apresenta uma qualidade visual bem melhor do que a imagem reconstruída da Figura 5.13, com o ônus de uma diminuição na taxa de compressão. A PSNR alcançada foi de 30,17 dB a 0,5 bpp, obtendo-se uma taxa de compressão de 16:1.

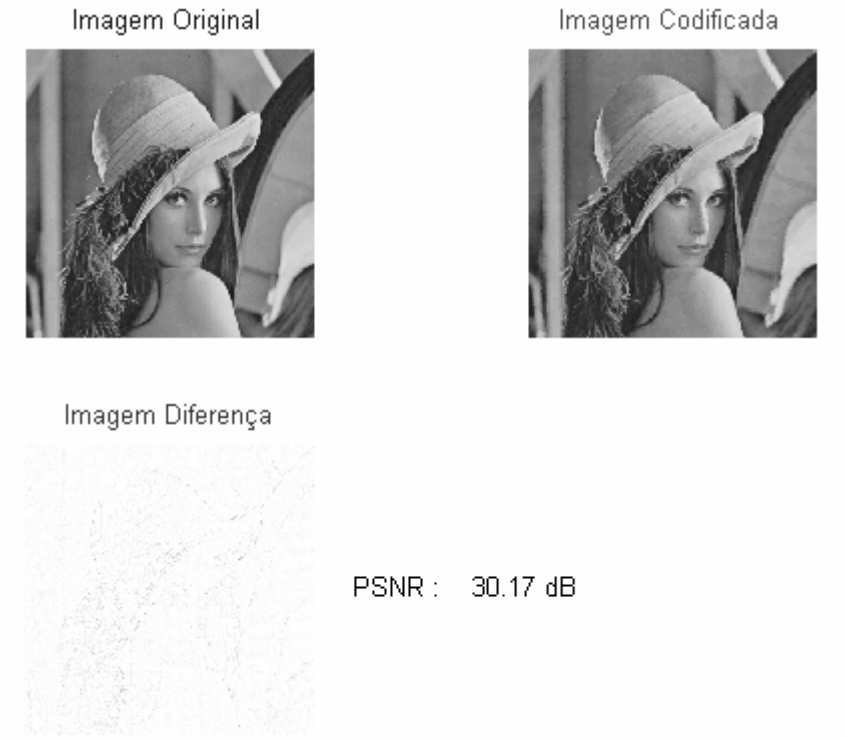


Figura 5.14– Resultados obtidos da VQ a 0,5 bpp.

5.8- Considerações Finais deste Capítulo

Neste capítulo foram apresentados os conceitos básicos de quantização vetorial (VQ), bem como uma visão geral das técnicas de quantização mais conhecidas, e alguns resultados práticos que a utilizam na codificação de imagens digitais. O algoritmo LBG apresentado na Seção 5.4 deste capítulo, considerado muito eficiente, é utilizado com muita freqüência na construção de *codebooks.*

No próximo capítulo é apresentado o modelo de compressão de imagens proposto neste trabalho que utiliza uma combinação da codificação fractal com a quantização vetorial.

CAPÍTULO VI

O MODELO DE CODIFICAÇÃO FRACTAL-VQ PROPOSTO NESTE TRABALHO

6.1- Introdução

A codificação fractal de imagens evoluiu consideravelmente desde a sua criação. O primeiro codificador apresentado utilizava o método da força bruta e demorava horas e horas para codificar uma simples imagem. Posteriormente, Fisher desenvolveu um codificador fractal acelerado que revolucionou a técnica, permitindo codificar imagens em segundos. A partir disso, muitos modelos de codificação, baseados na teoria fractal, foram desenvolvidos, a maioria combinando fractais com alguma outra técnica de codificação. Neste trabalho, o modelo proposto, referenciado neste trabalho como Fractal-VQ, é um codificador híbrido que utiliza a codificação fractal e a quantização vetorial (VQ), visando proporcionar um bom custo-benefício.

O maior obstáculo existente, no processo da codificação fractal, consiste no grande esforço computacional exigido, devido ao grande número de comparações necessário para associar um *range-block* a um *domain-block* do *domain pool*, motivo pelo qual a grande maioria das pesquisas referentes a esse assunto concentra-se em projetar codificadores que acelerem tal processo. O modelo de codificação Fractal-VQ proposto é descrito ao longo deste capítulo.

6.2- Descrição Geral do Modelo Fractal-VQ

Na codificação fractal pura, o conjunto de *domain-blocks* (*domain-pool*), é composto por blocos da própria imagem a ser codificada, sendo uma espécie de um *codebook* virtual, de forma que, obviamente, é preciso designar um *domain-pool* para cada imagem a ser codificada. No modelo proposto neste trabalho, uma base de imagens de treinamento é utilizada para gerar um *codebook* genérico utilizando-se o algoritmo LBG, descrito no Capítulo 4, com isso, o *domain-pool* restringe-se a um *codebook* de tamanho reduzido, dependendo da quantidade de níveis escolhida durante a execução do LBG. Neste trabalho, são utilizados nas simulações 3 (três) *codebooks*, um de 128 elementos (níveis) contendo vetores 4 x 4, outro com 96 níveis contendo vetores 8 x 8 e ainda, um outro com 32 elementos contendo vetores 16 x 16, totalizando 256 vetores-padrão de codificação. A Figura 6.1 ilustra o primeiro passo da implementação do codificador fractal-VQ, que é a construção dos *codebooks*.

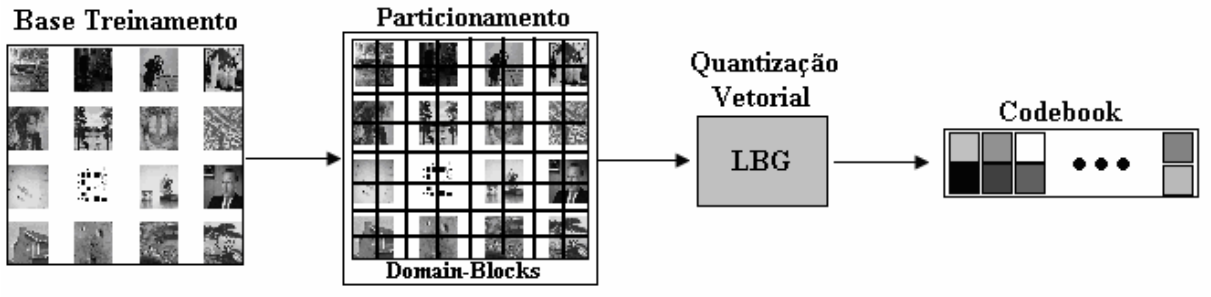


Figura 6.1- Construção dos *codebooks* utilizando-se o algoritmo LBG.

Para gerar cada um dos *codebooks* foram executados esses passos, primeiramente a base de treinamento é particionada em blocos quadrados, posteriormente o algoritmo LBG é executado originando um *codebook* de *n* níveis. Finalmente, cada um dos vetores do *codebook*

é classificado mediante a variância (*V*) e a média (*A*), conforme descrito na Seção 4.4.1 deste trabalho.

O processo de codificação, para qualquer imagem, é realizado semelhante à codificação fractal pura, com a diferença de que o *domain-pool* é o conjunto de *codebooks* designado pelo algoritmo LBG. Outra diferença é que a posição do bloco, que antes era representada por dois campos, agora é substituída apenas pelo índice do vetor no *codebook* que apresenta menor *rms* em relação ao vetor imagem que está sendo codificado. Uma ilustração da etapa de codificação do codificador proposto neste trabalho é apresentada na Figura 6.2.

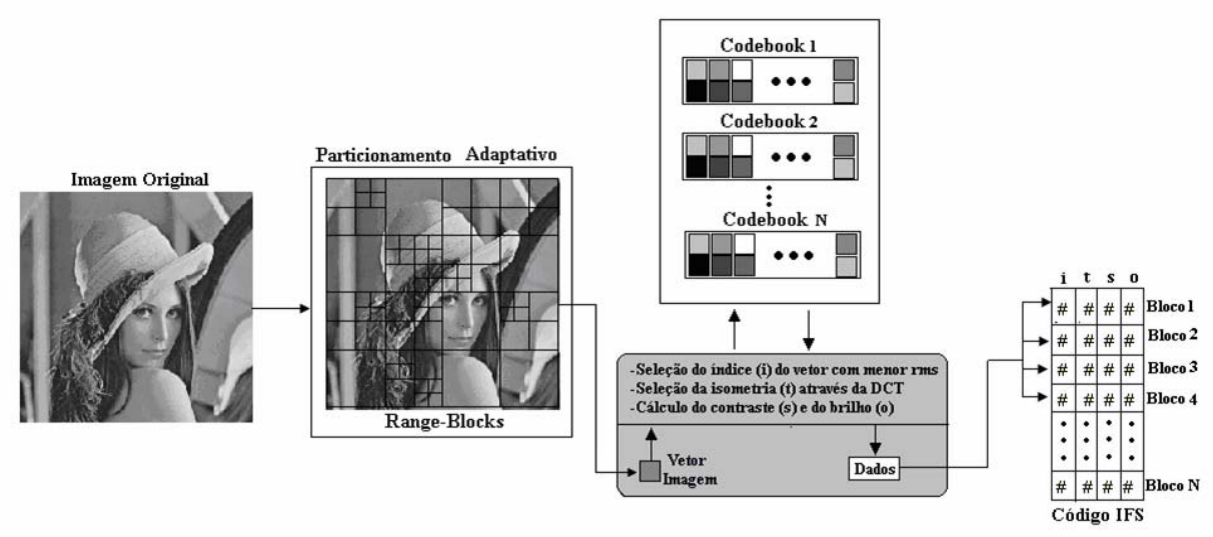


Figura 6.2- Codificação Fractal-VQ.

O primeiro passo é a divisão da imagem em blocos quadrados (*range-blocks*) utilizando o esquema *quadtree* de particionamento, descrito na Seção 4.2.1 deste trabalho, onde foram utilizados blocos de 4 x 4 (nível máximo de quadtree igual a 7), 8 x 8 e 16 x 16 (nível mínimo de quadtree igual a 5). Em seguida, cada bloco (*range-block*) é classificado

baseado na variância (*V*) e na média (*A*), conforme descrito na Seção 4.4.1, e depois comparado com os vetores da mesma classe do *codebook* correspondente.

Um outro ponto bastante importante no modelo em questão refere-se à seleção da isometria correspondente a cada vetor no momento da codificação, onde utiliza-se um método de aceleração baseado no produto interno da DCT, que será descrito na Seção 6.2.1 deste capítulo. O cálculo do brilho e contraste é realizado analogamente ao descrito na Seção 4.5 do capítulo 4 deste trabalho.

A taxa de bits é calculada da seguinte forma [6]:

Índice *i* do vetor candidato: 8 bits;

Nível de quadtree: 3 bits;

Isometria *t*: 3 bits;

Fator de contraste *s*: 5 bits;

Luminância *o*: 7 bits.

Totalizando 26 bits para cada bloco a ser codificado.

Na decodificação, o processo é semelhante ao de um decodificador fractal puro, entretanto, a imagem inicial é composta pelo conjunto de *codebooks*, que fornece o vetor candidato *i* escolhido na codificação. Após a seleção do vetor, é realizada a contração e em seguida a aplicação da isometria e ajuste do contraste e brilho, obtendo-se o vetor decodificado, que é parte integrante da imagem reconstruída. A Figura 6.3 mostra uma representação da decodificação Fractal-VQ.

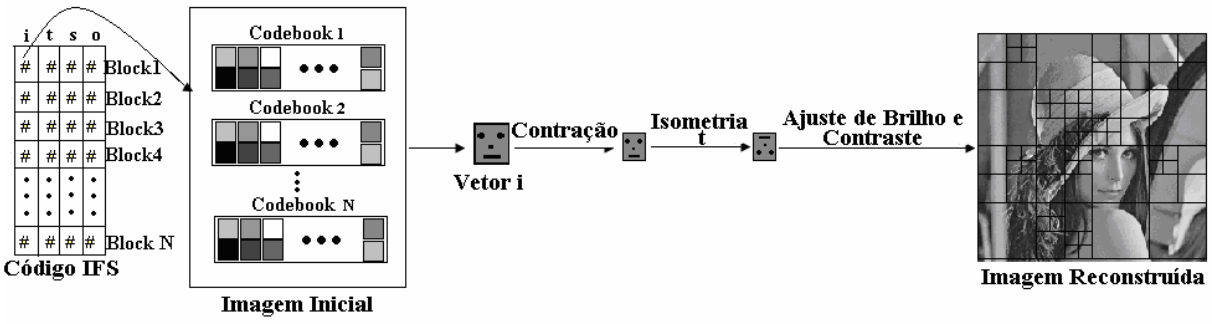


Figura 6.3- Decodificação Fractal-VQ.

Por tratar-se de um modelo híbrido, que engloba características da codificação fractal e da quantização vetorial, o processo de decodificação não é iterativo, como ocorre nos modelos de compressão fractais puros, apenas uma iteração é necessária para reconstruir a imagem codificada.

6.2.1- Classificação das Isometrias no Domínio da Freqüência

Nesta seção é apresentado um método de classificação para as isometrias utilizado por Truong e outros [58, 59], não sendo mais necessário armazená-las no *domain-pool* e, consequentemente, reduzindo-o oito vezes em relação ao tamanho inicial. Tal processo é baseado na transformada discreta Cosseno (DCT) e suas propriedades, ferramenta muito explorada em compressão de imagens digitais.

6.2.1.1- Transformada Discreta Cosseno (DCT)

A DCT – 2D, *F (i, j),* de um bloco de imagem *f (u, v)* de tamanho N x N, é definida por [16]:

$$F(i,\ j) = c(i)\,.\,c(j)\ \sum_{u=0}^{N-1}\ \sum_{v=0}^{N-1}\ f\ (u,\ v)\ \cos\left[\frac{(2u+1)\,i\,\pi}{2N}\right]\ \cos\left[\frac{(2v+1)\ j\,\pi}{2N}\right] \qquad (6.1)$$

para *u, v* = 0, 1, 2, . . ., *N*-1,

e a transformada inversa é obtida por:

$$f\ (u, v) = \sum_{i=0}^{N-1}\ \sum_{j=0}^{N-1}\ c(i)\,.\,c(j)\ F\ (i,\ j)\ \cos\left[\frac{(2u+1)\,i\,\pi}{2N}\right]\ \cos\left[\frac{(2v+1)\ j\,\pi}{2N}\right] \qquad (6.2)$$

para *x, y* = 0, 1, 2, . . ., *N*-1

onde *c* é obtido por

$$c(i) = \begin{cases} \dfrac{1}{\sqrt{2}}\ ; & \text{para } i = 0 \\ 1\ ; \quad \text{para } i = 1, 2, \ldots, N-1 \end{cases}$$

Considerando-se *f(u,v)* , *u,v = 0, 1,2 ..., N-1*, um bloco domínio (*domain-block*) *N x N* de um domain-pool , as oito orientações (isometrias) T_k, *k = 1, 2,3 , ..., 8,* de *f(u,v)* podem ser

representadas no plano cartesiano, conforme mostra a Figura 6.4, adaptada de Truong, Jeng Reed, Lee e Li [55]:

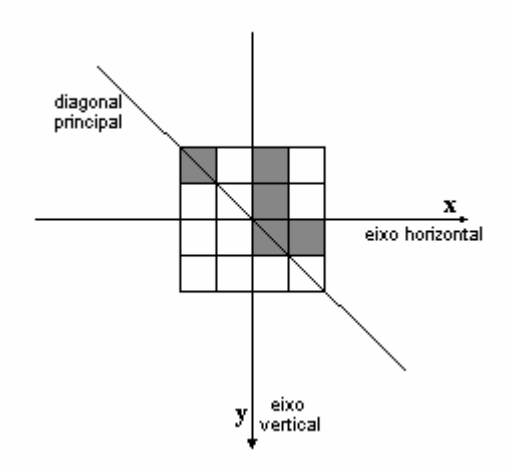


Figura 6.4- Bloco domínio (*domain-block*) no plano cartesiano.

1. Identidade: $T_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
2. Reflexão ortogonal em relação ao eixo médio horizontal do bloco: $T_2 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$
3. Reflexão ortogonal em relação ao eixo médio vertical do bloco: $T_3 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
4. Rotação de 180^0 (sentido horário) em torno do centro do bloco: $T_4 = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$
5. Reflexão ortogonal em relação à diagonal principal do bloco (eixo y = x): $T_5 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
6. Rotação de 90^0 (sentido horário) em torno do centro do bloco: $T_6 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$
7. Rotação de 270^0 (sentido horário) em torno do centro do bloco: $T_7 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$

8. Reflexão ortogonal em relação à diagonal secundária do bloco: $T_8 = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$

Seja $g_k(m,n)$, $k = 1, 2,3 , ..., 8$, a matriz que representa o resultado da aplicação de T_k sobre $f(u,v)$, ou seja, $g_k = T_k \cdot f(u,v), k = 1,2,...,8$, as representações matriciais para cada g_k são obtidas por:

1. Identidade: $g_1(m,n) = f(u,v)$ e a DCT de $g_1(m,n)$, denotada por $G_1(x,y)$, é definida como:

$$G_1(x, y) = c(x) \cdot c(y) \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} f(u, v) \cos\left[\frac{(2u+1)\, x\, \pi}{2N}\right] \cos\left[\frac{(2v+1)\, y\, \pi}{2N}\right] \quad (6.3)$$

da Equação (6.1) tem-se que:

$$G_1(x, y) = F(i, j) \quad (6.4)$$

2. Reflexão ortogonal em relação ao eixo médio horizontal do bloco, ou seja, em relação ao eixo $u = (N-1)/2$: a mudança de coordenadas é obtida pela reflexão do bloco original em relação ao eixo $u = (N-1)/2$, dessa forma, a nova coordenada (m,n) é obtida por:

$$\begin{bmatrix} m - \frac{N-1}{2} \\ n \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} u - \frac{N-1}{2} \\ v \end{bmatrix} \tag{6.5}$$

resultando em $m = -u + N - 1$ e $n = v$, portanto

$$g_2(m,n) = f(-u+N-1, v)$$

A DCT $G_2(x,y)$ de $g_2(m,n)$ é então escrita como:

$$\begin{aligned} G_2(x, y) &= c(x) \cdot c(y) \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} g_2(m, n) \cos\left[\frac{(2m+1)\, x\, \pi}{2N}\right] \\ &\quad \cdot \cos\left[\frac{(2n+1)\, y\, \pi}{2N}\right] \\ &= c(x) \cdot c(y) \sum_{-u+N-1=0}^{N-1} \sum_{v=0}^{N-1} f(u, v) \cos\left[\frac{(2(-u+N-1)+1)\, x\, \pi}{2N}\right] \\ &\quad \cdot \cos\left[\frac{(2v+1)\, y\pi}{2N}\right] \\ &= c(x) \cdot c(y) \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} f(u, v) \cos\left[\frac{(2u+1)\, x\pi}{2N} + x\pi\right] \\ &\quad \cdot \cos\left[\frac{(2v+1)\, y\pi}{2N}\right] \\ &= (-1)^x c(x) \cdot c(y) \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} f(u, v) \cos\left[\frac{(2u+1)\, x\pi}{2N}\right] \\ &\quad \cdot \cos\left[\frac{(2v+1)\, y\pi}{2N}\right] \\ G_2(x, y) &= (-1)^x F(i, j) \end{aligned} \tag{6.6}$$

3. Reflexão ortogonal em relação ao eixo médio vertical do bloco, ou seja, em relação ao eixo $v = (N-1)/2$: a mudança de coordenadas é obtida pela reflexão do bloco original em relação ao eixo $v = (N-1)/2$, dessa forma, a nova coordenada (m,n) é obtida por:

$$\begin{bmatrix} m \\ n - \frac{N-1}{2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} u \\ v - \frac{N-1}{2} \end{bmatrix} \tag{6.7}$$

resultando em $m = u$ e $n = -v+N-1$, portanto

$$g_3(m,n) = f(u,-v+N-1)$$

A DCT $G_3(x,y)$ de $g_3(m,n)$ é então calculada de maneira análoga à Equação (6.6), da seguinte forma:

$$\begin{aligned} G_3(x, y) &= c(x) \cdot c(y) \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} g_3(m, n) \cos\left[\frac{(2m+1)\, x\, \pi}{2N}\right] \\ &\qquad \cdot \cos\left[\frac{(2n+1)\, y\, \pi}{2N}\right] \\ &= c(x) \cdot c(y) \sum_{u=0}^{N-1} \sum_{-v+N-1=0}^{N-1} f(u, v) \cos\left[\frac{(2u+1)\, x\, \pi}{2N}\right] \\ &\qquad \cdot \cos\left[\frac{(2(-v+N-1)+1)\, y\pi}{2N}\right] \\ &= c(x) \cdot c(y) \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} f(u, v) \cos\left[\frac{(2u+1)\, x\pi}{2N}\right] \\ &\qquad \cdot \cos\left[\frac{(2v+1)\, y\pi}{2N} + y\pi\right] \\ &= (-1)^y\, c(x) \cdot c(y) \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} f(u, v) \cos\left[\frac{(2u+1)\, x\pi}{2N}\right] \\ &\qquad \cdot \cos\left[\frac{(2v+1)\, y\pi}{2N}\right] \end{aligned}$$

$$G_3(x, y) = (-1)^y F(i, j) \tag{6.8}$$

4. Rotação de 180^0 (sentido horário) em torno do centro do bloco: a mudança de coordenadas é obtida pela reflexão do bloco original em relação ao eixo u = *(N-1)/2* e *v = (N-1)/2*, dessa forma, a nova coordenada *(m,n)* é obtida por:

$$\begin{bmatrix} m-\frac{N-1}{2} \\ n-\frac{N-1}{2} \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} u-\frac{N-1}{2} \\ v-\frac{N-1}{2} \end{bmatrix} \tag{6.9}$$

resultando em $m = -u+N-1$ e $n = -v+N-1$, portanto

$$g_4(m,n) = f(-u+N-1,-v+N-1)$$

A DCT é então calculada por:

$$G_4(x, y) = c(x).c(y) \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} g_4(m, n) \cos\left[\frac{(2m+1)\, x\, \pi}{2N}\right] \cdot \cos\left[\frac{(2n+1)\, y\, \pi}{2N}\right]$$

$$= c(x).c(y) \sum_{-u+N-1=0}^{N-1} \sum_{-v+N-1=0}^{N-1} f(u, v) \cos\left[\frac{(2(-u+N-1)+1)\, x\, \pi}{2N}\right] \cdot \cos\left[\frac{(2(-v+N-1)+1)\, y\pi}{2N}\right]$$

$$= c(x).c(y) \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} f(u, v) \cos\left[\frac{(2u+1)\, x\pi}{2N} + x\pi\right] \cdot \cos\left[\frac{(2v+1)\, y\pi}{2N} + y\pi\right]$$

$$= (-1)^{x+y}\, c(x).c(y) \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} f(u, v) \cos\left[\frac{(2u+1)\, x\pi}{2N}\right] \cdot \cos\left[\frac{(2v+1)\, y\pi}{2N}\right]$$

$$G_4(x, y) = (-1)^{x+y} F(i, j) \tag{6.10}$$

5. Reflexão ortogonal em relação à diagonal principal do bloco (eixo $y = x$ no plano): a mudança de coordenadas é definida da seguinte forma:

$$\begin{bmatrix} m \\ n \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} u \\ v \end{bmatrix} \tag{6.11}$$

resultando em $m = v$ e $n = u$, assim:

$$g_5(m,n) = f(v,u)$$

e da mesma forma:

$$G_5(x, y) = c(x) \cdot c(y) \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} g_5(m, n) \cos\left[\frac{(2m+1)\, x\, \pi}{2N}\right] \cdot \cos\left[\frac{(2n+1)\, y\, \pi}{2N}\right] =$$

$$= c(x) \cdot c(y) \sum_{v=0}^{N-1} \sum_{u=0}^{N-1} f(u, v) \cos\left[\frac{(2v+1)\, x\, \pi}{2N}\right] \cdot \cos\left[\frac{(2u+1)\, y\pi}{2N}\right]$$

$$= c(y) \cdot c(x) \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} f(u, v) \cos\left[\frac{(2u+1)\, y\, \pi}{2N}\right] \cdot \cos\left[\frac{(2v+1)\, x\pi}{2N}\right] \tag{6.12}$$

$$G_5(x, y) = F(j, i)$$

6. Rotação de 90^0 (sentido horário) em torno do centro do bloco: a mudança de coordenadas é obtida por:

$$\begin{bmatrix} m - \dfrac{N-1}{2} \\ n \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} u \\ v - \dfrac{N-1}{2} \end{bmatrix} \tag{6.13}$$

e assim, $m = -v+N-1$ e $n = u$, então $g_6(m,n) = f(-v+N-1,u)$ *e a DCT* $G_6(x,y)$ é obtida por:

$$\begin{aligned} &= \mathrm{c}(y) \,.\, \mathrm{c}(x) \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} f(u, v) \cos\left[\frac{(2u+1)\, y\pi}{2N} + y\pi\right] \\ &\qquad \cdot \cos\left[\frac{(2v+1)\, x\pi}{2N}\right] \\ &= (-1)^y \, \mathrm{c}(y) \,.\, \mathrm{c}(x) \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} f(u, v) \cos\left[\frac{(2u+1)\, y\pi}{2N}\right] \\ &\qquad \cdot \cos\left[\frac{(2v+1)\, x\pi}{2N}\right] \end{aligned} \tag{6.14}$$

$$G_6(x, y) = (-1)^y F(j, i)$$

7. Rotação de 270^0 (no sentido horário) em torno do centro do bloco: neste caso, tem-se a seguinte mudança de coordenadas:

$$\begin{bmatrix} m \\ n - \dfrac{N-1}{2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} u - \dfrac{N-1}{2} \\ v \end{bmatrix} \tag{6.15}$$

dessa forma, $m = v$ e $n = -u+N-1$ e $g_7(m,n) = f(v,-u+N-1)$ *e a DCT* $G_7(x,y)$ é obtida da seguinte forma:

$$G_7(x,y) = \frac{2}{N} c(x).c(y) \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} g_7(m,n) \cos\left[\frac{(2m+1)\,x\,\pi}{2N}\right] \cdot \cos\left[\frac{(2n+1)\,y\,\pi}{2N}\right]$$

$$= \frac{2}{N} c(x).c(y) \sum_{v=0}^{N-1} \sum_{-u+N-1=0}^{N-1} f(u,v) \cos\left[\frac{(2v+1)\,x\,\pi}{2N}\right] \cdot \cos\left[\frac{(2(-u+N-1)+1)\,y\pi}{2N}\right]$$

$$= \frac{2}{N} c(y).c(x) \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} f(u,v) \cos\left[\frac{(2u+1)\,y\pi}{2N}\right] \cdot \cos\left[\frac{(2v+1)\,x\pi}{2N} + x\pi\right]$$

$$= (-1)^x \frac{2}{N} c(y).c(x) \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} f(u,v) \cos\left[\frac{(2u+1)\,y\pi}{2N}\right] \cdot \cos\left[\frac{(2v+1)\,x\pi}{2N}\right] \qquad (6.16)$$

$$G_7(x,y) = (-1)^x F(j,i)$$

8. Reflexão ortogonal em relação à diagonal secundária do bloco: tem-se a seguinte mudança de coordenadas:

$$\begin{bmatrix} m - \frac{N-1}{2} \\ n - \frac{N-1}{2} \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} u - \frac{N-1}{2} \\ v - \frac{N-1}{2} \end{bmatrix} \qquad (6.17)$$

dessa forma resulta que $m = -v+N-1$ e $n = -u+N-1$, portanto:

$g_8(m,n) = f(-v+N-1,-u+n-1)$ e a DCT G_8 é obtida por:

$$G_8(x, y) = \frac{2}{N} c(x) \cdot c(y) \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} g_8(m, n) \cos\left[\frac{(2m+1)\, x\, \pi}{2N}\right] \cdot \cos\left[\frac{(2n+1)\, y\, \pi}{2N}\right]$$

$$= \frac{2}{N} c(x) \cdot c(y) \sum_{-v+N-1=0}^{N-1} \sum_{-u+N-1=0}^{N-1} f(v, u) \cos\left[\frac{(2(-v+N-1)+1)\, x\, \pi}{2N}\right] \cdot \cos\left[\frac{(2(-u+N-1)+1)\, y\pi}{2N}\right]$$

$$G_8(x, y) = \frac{2}{N} c(y) \cdot c(x) \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} f(u, v) \cos\left[\frac{(2u+1)\, y\pi}{2N} + y\pi\right] \cdot \cos\left[\frac{(2v+1)\, x\pi}{2N} + x\pi\right]$$

$$= (-1)^{y+x} \frac{2}{N} c(y) \cdot c(x) \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} f(u, v) \cos\left[\frac{(2u+1)\, y\pi}{2N}\right] \cdot \cos\left[\frac{(2v+1)\, x\pi}{2N}\right] \quad (6.18)$$

$$G_8(x, y) = (-1)^{x+y} F(j, i)$$

Conforme demonstrado, dado um bloco de imagem f_k, $k = 1, 2, ..., 8$, onde k representa as oito orientações referidas e seja F_k a DCT de f_k, pode-se então concluir que [58]:

$$F_2(m,n) = (-1)^m F_1(m,n)$$
$$F_3(m,n) = (-1)^n F_1(m,n) \quad (6.19)$$
$$F_4(m,n) = (-1)^{m+n} F_1(m,n)$$

e

$$F_5(m,n) = F_1(n,m)$$
$$F_6(m,n) = F_2(n,m) = (-1)^n F_1(n,m)$$
$$F_7(m,n) = F_3(n,m) = (-1)^m F_1(n,m) \quad (6.20)$$
$$F_8(m,n) = F_4(n,m) = (-1)^{m+n} F_1(n,m)$$

Para ilustrar na prática os resultados apresentados nas Equações (6.19) e (6.20), seja f uma matriz definida por $f = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ e sua DCT F resultante $F = \begin{bmatrix} 5 & -1 \\ -2 & 0 \end{bmatrix}$, tem-se as seguintes matrizes correspondentes:

$$f_1 = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \text{ e } F_1 = \begin{bmatrix} 5 & -1 \\ -2 & 0 \end{bmatrix}$$

$$f_2 = \begin{bmatrix} 3 & 4 \\ 1 & 2 \end{bmatrix} \text{ e } F_2 = \begin{bmatrix} 5 & -1 \\ 2 & 0 \end{bmatrix} = (-1)^m F_1(m,n)$$

$$f_3 = \begin{bmatrix} 2 & 1 \\ 4 & 3 \end{bmatrix} \text{ e } F_3 = \begin{bmatrix} 5 & 1 \\ -2 & 0 \end{bmatrix} = (-1)^n F_1(m,n)$$

$$f_4 = \begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix} \text{ e } F_4 = \begin{bmatrix} 5 & 1 \\ 2 & 0 \end{bmatrix} = (-1)^{m+n} F_1(m,n)$$

$$f_5 = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \text{ e } F_5 = \begin{bmatrix} 5 & -2 \\ -1 & 0 \end{bmatrix} = F_1(n,m)$$

$$f_6 = \begin{bmatrix} 3 & 1 \\ 4 & 2 \end{bmatrix} \text{ e } F_6 = \begin{bmatrix} 5 & 2 \\ -1 & 0 \end{bmatrix} = (-1)^n F_1(n,m)$$

$$f_7 = \begin{bmatrix} 2 & 4 \\ 1 & 3 \end{bmatrix} \text{ e } F_7 = \begin{bmatrix} 5 & -2 \\ 1 & 0 \end{bmatrix} = (-1)^m F_1(n,m)$$

$$f_8 = \begin{bmatrix} 4 & 2 \\ 3 & 1 \end{bmatrix} \text{ e } F_8 = \begin{bmatrix} 5 & 2 \\ 1 & 0 \end{bmatrix} = (-1)^{n+m} F_1(n,m)$$

Pode-se observar, que para um determinado *domain-block* *f*, as oito isometrias podem ser obtidas pela sua DCT, visto que as transformadas das oito orientações diferem entre si, basicamente, pela mudança de coordenadas. Tal propriedade pode ser explorada para eliminar cálculos repetitivos, reduzindo o tempo de processamento consideravelmente.

6.2.1.2- Classificação das Isometrias pelo Produto Interno da DCT

A codificação fractal para uma imagem monocromática *f(x,y)*, é representada por uma transformação afim da seguinte forma [58]:

$$\Phi\begin{bmatrix} x \\ y \\ f(x,y) \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ 0 & 0 & p \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ f(x,y) \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ q \end{bmatrix} \tag{6.21}$$

onde:

submatriz 2 x 2 $\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ - contração no plano x-y e isometria;

subvetor 2-D (t_x,t_y) - translação no plano x-y;

p - contraste;

q - brilho;

Para uma imagem digital de 512x512 pixels, utilizando-se *domain-blocks* com tamanhos quadrados de 16 x 16 pixels, com sobreposição de 50 %, resulta num *domain-pool* de (512/8 -1) x (512/8 - 1) = 3969 elementos. Para *range-blocks* de tamanhos 8 x 8 pixels, não sobrepostos, tem-se um conjunto de 4096 blocos e considerando-se as oito isometrias para cada bloco domínio e um fator de contração, neste exemplo, igual a ½, a submatriz 2 x 2 da Equação (6.21) para cada bloco a ser codificado é uma dentre as seguintes:

$$T_1 = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix};\ T_2 = \begin{bmatrix} -\frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix};\ T_3 = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & -\frac{1}{2} \end{bmatrix};\ T_4 = \begin{bmatrix} -\frac{1}{2} & 0 \\ 0 & -\frac{1}{2} \end{bmatrix};$$

$$T_5 = \begin{bmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & 0 \end{bmatrix};\ T_6 = \begin{bmatrix} 0 & -\frac{1}{2} \\ \frac{1}{2} & 0 \end{bmatrix};\ T_7 = \begin{bmatrix} 0 & \frac{1}{2} \\ -\frac{1}{2} & 0 \end{bmatrix};\ T_8 = \begin{bmatrix} 0 & -\frac{1}{2} \\ -\frac{1}{2} & 0 \end{bmatrix};$$

O processo de codificação resume-se em encontrar um bloco *f* no *domain-pool* e a transformação afim Φ que mais se aproxima de um determinado *range-block g*, isto ocorre quando o MSE $\varepsilon^2 = \left|\Phi(f) - g\right|^2$ é o mínimo possível, que significa minimizar o resultado da Equação (6.22).

$$\varepsilon^2 = \sum_{i=0}^{7}\sum_{j=0}^{7}((p \cdot f(i,j) + q) - g(i,j))^2 \tag{6.22}$$

O valor de *p* para cada *f* e suas oito orientações também pode ser calculado pelo produto interno da Equação (6.23).

$$p = \frac{\left\langle \hat{f}_k, \hat{g} \right\rangle}{\left\langle \hat{f}_k, \hat{f}_k \right\rangle} \tag{6.23}$$

onde:

$\left\langle \hat{f}_k, \hat{g} \right\rangle$ - produto interno de $\hat{f}_k$ e $\hat{g}$;

$\hat{f} = f - m_f$;

$\hat{g} = g - m_g$;

m_f - valor médio do bloco f;

m_g - valor médio de g;

e

$$q = m_g - p\, m_f \tag{6.24}$$

a partir disso, da Equação (6.22) pode-se obter:

$$\varepsilon^2 = \langle \hat{g}, \hat{g} \rangle - \frac{p}{\left\langle \hat{f}_k, \hat{g} \right\rangle} = \langle \hat{g}, \hat{g} \rangle - \frac{\left\langle \hat{f}_k, \hat{g} \right\rangle^2}{\left\langle \hat{f}_k, \hat{f}_k \right\rangle} \tag{6.25}$$

Na Equação (6.25) verifica-se que ε^2 é mínimo quando o produto interno $\left\langle \hat{f}_k, \hat{g} \right\rangle$ é máximo possível. Sendo $\hat{F}_k$ e $\hat{G}$ a transformada DCT de $\hat{f}_k$ e $\hat{g}$ respectivamente, para *k = 1, 2, ...,8*, o produto interno $\left\langle \hat{f}_k, \hat{g} \right\rangle$ também pode ser obtido no domínio da DCT como segue:

$$\delta_k = \left\langle \hat{f}_k, \hat{g} \right\rangle = \left\langle \hat{F}_k, \hat{G} \right\rangle = \sum_{i=0}^{7} \sum_{j=0}^{7} \hat{F}_k(i, j) \cdot \hat{G}(i, j) \tag{6.26}$$

Para k =1,

$$\delta_1 = \sum_{i=0}^{7} \sum_{j=0}^{7} \hat{F}_1(i, j) \cdot \hat{G}(i, j) = \sum_{i=0}^{7} \sum_{j=0}^{7} a_{ij} \tag{6.27}$$

onde $a_{ij} = \hat{F}_1(i, j) \cdot \hat{G}(i, j)$. Da Equação (6.19) pode-se obter que:

$$\delta_1 = \sum_{i=0}^{7}\sum_{j=0}^{7} a_{ij}$$

$$\delta_2 = \sum_{i=0}^{7}\sum_{j=0}^{7} (-1)^i a_{ij} \;;$$

$$\delta_3 = \sum_{i=0}^{7}\sum_{j=0}^{7} (-1)^j a_{ij} \;; \qquad (6.28)$$

$$\delta_4 = \sum_{i=0}^{7}\sum_{j=0}^{7} (-1)^{i+j} a_{ij}$$

e

$$\delta_5 = \sum_{i=0}^{7}\sum_{j=0}^{7} \hat{F}_5(i,j)\cdot\hat{G}(i,j) = \sum_{i=0}^{7}\sum_{j=0}^{7} \hat{F}_1(j,i)\cdot\hat{G}(i,j) = \sum_{i=0}^{7}\sum_{j=0}^{7} b_{ij}$$

onde $b_{ij} = \hat{F}_1(j,i)\cdot\hat{G}(i,j)$ e da Equação (6.20) tem-se que:

$$\delta_5 = \sum_{i=0}^{7}\sum_{j=0}^{7} b_{ij}$$

$$\delta_6 = \sum_{i=0}^{7}\sum_{j=0}^{7} (-1)^j b_{ij} \;;$$

$$\delta_7 = \sum_{i=0}^{7}\sum_{j=0}^{7} (-1)^i b_{ij} \;; \qquad (6.29)$$

$$\delta_8 = \sum_{i=0}^{7}\sum_{j=0}^{7} (-1)^{j+i} b_{ij}$$

Finalmente, para encontrar a isometria (orientação) *k* mais adequada basta selecionar o δ_k máximo, ou seja:

$$k = \max(\delta_1, \delta_2, ..., \delta_8) \qquad (6.30)$$

Observa-se que no domínio da freqüência existe uma redundância no cálculo de δ_k, *k = 1,2,...,8*, envolvendo basicamente a obtenção de a_{ij} e b_{ij} ,que não acontece no domínio

espacial. O tempo de processamento gasto para encontrar *k*, utilizando-se as Equações (6.27) a (6.30), é bem inferior ao tempo gasto nas comparações entre o *range-block* e as oito simetrias do *domain-block* que está sendo analisado, como é realizado no algoritmo que utiliza o método da força bruta. Na próxima seção é apresentado um algoritmo que contribui para diminuir ainda mais esse tempo de processamento.

6.2.1.3- Algoritmo de Classificação

Na Seção 6.2.1 foi mostrado que a busca por uma isometria *k, k=1,2,...8,* de um *domain-block f,* que melhor se associa a um determinado *range-block g* pode ser reduzida ao cômputo dos produtos internos DCT dos grupos de Equações (6.28) e (6.29) e pode-se verificar ainda, que eles podem ser calculados simultaneamente. Nesta subseção é apresentado um algoritmo simples para classificação baseado na comparação das baixas freqüências horizontais e verticais [59], no qual apenas um dos grupos de Equações (6.28 ou 6.29) precisa ser calculado, reduzindo ainda mais o tempo de busca necessário para encontrar a isometria mais apropriada.

Sejam T_1, T_2, ..., T_8 as oito simetrias apresentadas na Seção 6.2.1.1 deste capítulo, dividindo-as em dois grupos H_1 e H_2 tal que $H_1 = \{T_1, T_2, T_3, T_4\}$ e $H_2 = \{T_5, T_6, T_7, T_8\}$ e sendo $f_1(i,j)$ um *domain-block* a ser analisado para um determinado *range-block* $g(i,j)$, então se $F_1(0,1) \geq F_1(1,0)$ e $G(0,1) \geq G(1,0)$, f_1 e g são blocos horizontais, e ainda, se $F_1(0,1) < F_1(1,0)$ e $G(0,1) < G(1,0)$ então f_1 e g são blocos verticais. Nestes dois casos as isometrias do grupo H_1 apresentam um menor MSE em relação às simetrias do grupo H_2 e, dessa forma, apenas o grupo de Equações (6.28) precisa ser computado. Para todos os outros casos, apenas as transformações do grupo H_2 são requeridas. Ou seja:

Se ($F_1(0,1) \geq F_1(1,0)$ e $G(0,1) \geq G(1,0)$) ou ($F_1(0,1) < F_1(1,0)$ e $G(0,1) < G(1,0)$)

Então $k = \max(\delta_1, \delta_2, \delta_3, \delta_4)$;

Senão $k = \max(\delta_5, \delta_6, \delta_7, \delta_8)$

onde F_1 e G são as transformadas DCT de f_1 e g respectivamente.

Pode-se verificar, claramente que com a utilização do produto interno da DCT pode-se acelerar a escolha da isometria referente ao mapeamento de cada *range-block* por um procedimento simples de classificação de blocos baseado nas amplitudes de freqüências horizontais e verticais, sem a necessidade de armazená-las no *domain-pool*. Embora o cálculo da DCT para cada *range-block* envolva um custo considerável, com a utilização da transformada rápida DCT os resultados comprovam a melhoria da eficiência sem alterar a qualidade da imagem codificada.

6.3- Considerações Finais deste Capítulo

Neste capítulo foram apresentadas as características da codificação Fractal-VQ, bem como a descrição geral do modelo proposto neste trabalho. Os mecanismos utilizados, tais como a utilização da DCT na escolha da transformação geométrica mais adequada e da construção de um *codebook* (*domain-pool*) genérico e bastante reduzido, comparado ao *domain-pool* de um codificador fractal puro, tornam o sistema de compressão Fractal-VQ uma boa alternativa na codificação de imagens digitais com redução substancial do esforço computacional necessário, comparado à codificação fractal pura, mantendo bons níveis de PSNR. No próximo capítulo são apresentados os resultados práticos e comparações entre o modelo de codificação Fractal-VQ proposto e outros codificadores de imagens digitais.

CAPÍTULO VII

RESULTADOS OBTIDOS

7.1- Introdução

Neste capítulo são mostrados os resultados práticos e as comparações com outros modelos de codificação, provenientes das simulações efetuadas utilizando-se o codificador Fractal-VQ proposto neste trabalho e outros codificadores de imagens presentes na literatura. Primeiramente, são apresentados os resultados de cada um dos codificadores estudados, sendo eles o codificador fractal puro (codificador de Fisher), o quantizador vetorial e o codificador Fractal-VQ, em seguida é realizada uma comparação entre os três codificadores em termos de PSNR e velocidade de processamento. Finalmente, é abordada uma comparação entre o modelo proposto e um outro codificador híbrido proposto por Iano, Silva e Cruz [12].

7.2- Codificador de Fisher

Fisher [6] apresentou um codificador acelerado que é uma referência no estudo da compressão fractal de imagens, por esse motivo, neste trabalho tal sistema foi utilizado como base de comparação para o modelo proposto. O esquema de particionamento utilizado por Fisher foi descrito na Seção 4.2.1 do capítulo IV deste trabalho, conhecido como *quadtree* e a estratégia de classificação de blocos é baseada na variância e na média para dividir os *range-blocks* em superclasses e subclasses, como foi descrito na Seção 4.4 do Capítulo IV deste

trabalho. Os cálculos do contraste *s* e da luminância *o*, bem como da distância *rms* utilizada e outros detalhes matemáticos foram mostrados na Seção 4.5 do capítulo IV deste trabalho. A Figura 7.1 mostra uma representação da codificação de imagens utilizando-se o modelo proposto por Fisher.

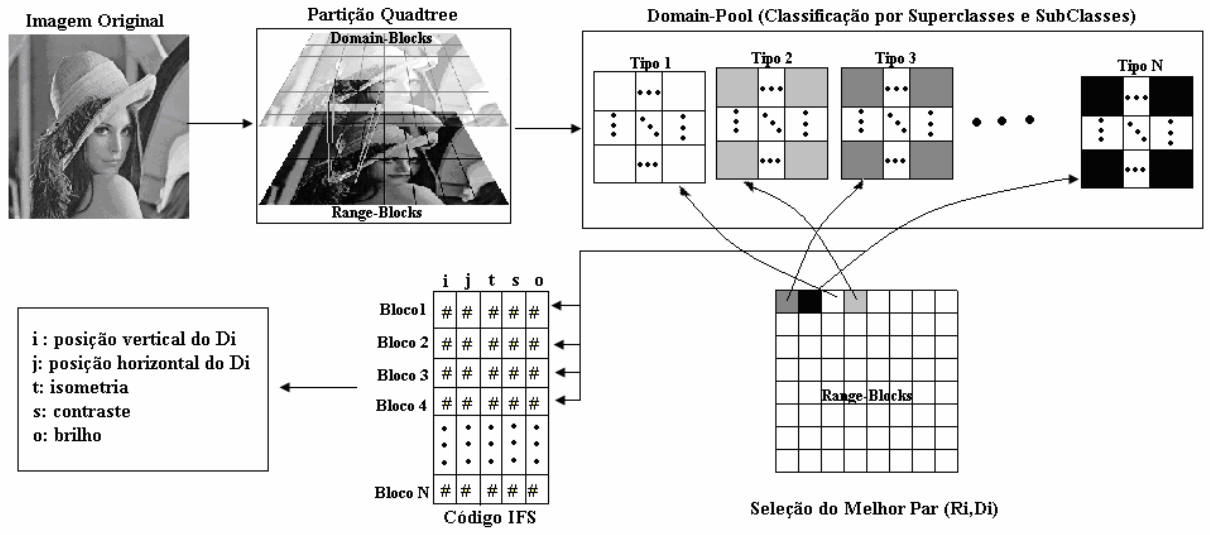


Figura 7.1- Codificador fractal puro de Fisher.

O número de bits utilizado para codificar cada quadro da imagem é distribuído da seguinte forma, levando-se em consideração as imagens de 512 x 512 pixels:

D_i (posição vertical do *domain-block*): 8 bits, considerando-se a sobreposição de 50%;

D_j (posição horizontal do *domain-block*): 8 bits, considerando-se a sobreposição de 50%;

Nível de *quadtree*: 3 bits;

Isometria: 3 bits;

Fator de contraste: 5 bits;

Luminância: 7 bits;

Pode-se verificar que para o codificador de Fisher são necessários essencialmente 34 bits para se codificar cada bloco da imagem.

7.3- Quantizador Vetorial

Um quantizador vetorial, cujos *codebooks* foram gerados pelo algoritmo LBG também servirá como base de comparação para o codificador proposto. As imagens utilizadas como base de treinamento foram apresentadas na Seção 5.7 do capítulo V deste trabalho. A taxa de bits está diretamente relacionada com as dimensões do bloco e o número de elementos presentes no *codebook*. Maiores detalhes sobre a técnica de quantização vetorial foram apresentados no capítulo V deste trabalho. A Figura 7.2 ilustra o funcionamento do quantizador vetorial implementado neste trabalho.

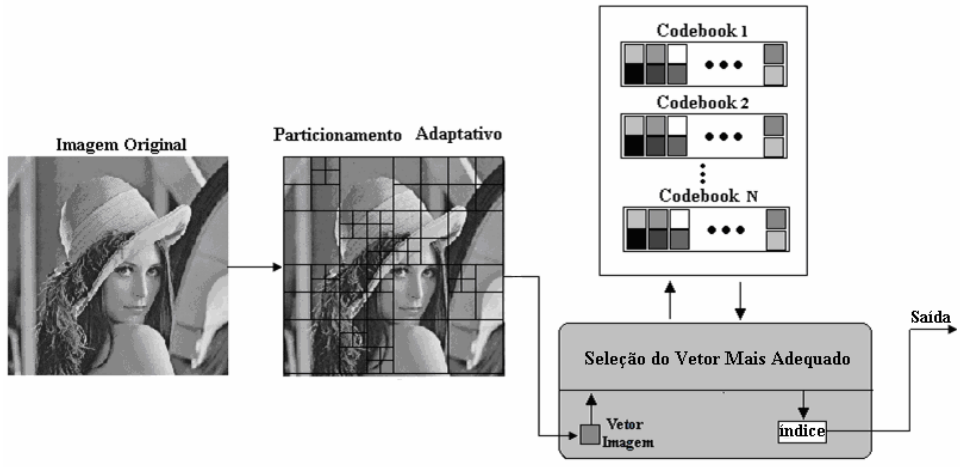


Figura 7.2- Quantizador vetorial.

O processo de codificação inicia-se com o particionamento *quadtree* da imagem de entrada, posteriormente cada bloco da imagem é comparado com os vetores do *codebook* correspondente às dimensões daquele bloco e então se transmite o índice do vetor selecionado

no *codebook*. A decodificação consiste em recuperar nos *codebooks* os vetores cujos índices foram transmitidos.

7.4- Resultados Obtidos

Nesta seção são apresentados alguns resultados obtidos na implementação dos três codificadores: codificador de Fisher, quantizador vetorial e o codificador Fractal-VQ proposto neste trabalho. A linguagem utilizada é o Matlab, versão 6.1 e o equipamento usado é um computador PC padrão equipado com processador Intel Pentium IV 3.33 MHz e 512 MB de memória RAM. As imagens escolhidas para as simulações são apresentadas na Figura 7.3, muito conhecidas no estudo do processamento de imagens.

(a) (b) (c)

Figura 7.3 – Imagens 512 x 512 pixels e 8 bpp; (a) Lena; (b) Couple; (c) Swan.

As imagens da Figura 7.3 foram selecionadas por apresentarem diferentes características entre si, permitindo uma melhor avaliação e comparação dos resultados.

7.4.1- Métricas de Fidelidade

Pela comparação com a imagem original é possível obter-se uma estimativa da qualidade da imagem reconstruída. As métricas mais conhecidas, embora simples e objetivas, é o erro quadrático médio (*MSE*) e a relação sinal-ruído de pico *(PSNR*).

O Erro Quadrático Médio é o quadrado do erro cumulativo entre a imagem original e a reconstruída, que é obtido por [16]:

$$MSE = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(x_{i,j} - \hat{x}_{i,j})^2 \tag{7.1}$$

onde:

N – tamanho da imagem*;*

$x_{i,j}$ - valor do pixel original na posição i,j;

$\hat{x}_{i,j}$ – valor do pixel reconstruído na posição i,j;

A relação Sinal-Ruído de Pico (PSNR) é obtida por [16]:

$$PSNR = 10 \cdot \log_{10} \frac{(2^n - 1)^2}{MSE}\, dB \tag{7.2}$$

onde:

n – número de bits por pixel;

Não existe uma boa correlação entre o *MSE*, *PSNR* e a percepção visual humana, entretanto, devido à falta de medidas precisas e pela ausência de padronização, elas são amplamente utilizadas.

7.4.2- Resultados Obtidos do Codificador de Fisher

Os resultados obtidos ao utilizar o codificador de Fisher para as imagens ilustradas na Figura 7.3 são mostrados nas Figuras 7.4 a 7.6. Pode-se observar que os resultados produzidos pelo codificador de Fisher são bastante satisfatórios, fornecendo imagens com boa qualidade visual, considerando-se a taxa de bits utilizada de 0,49 bpp, que proporciona uma taxa de compressão de aproximadamente 16:1.

(a)

(b)

Figura 7.4 – Resultado obtido da codificação da imagem Lena pelo codificador de Fisher à 0,49 bpp: (a) imagem original; (b) imagem reconstruída – PSNR = 32,33 dB.

(a)

(b)

Figura 7.5 – Resultado obtido da codificação da imagem Couple pelo codificador de Fisher à 0,49 bpp: (a) imagem original; (b) imagem reconstruída – PSNR = 28,41 dB.

(a)

(b)

Figura 7.6 – Resultado obtido da codificação da imagem Swan pelo codificador de Fisher à 0,49 bpp: (a) imagem original; (b) imagem reconstruída – PSNR = 26,50 dB.

A Tabela 7.1 contém um resumo dos resultados obtidos para o codificador de Fisher e os tempos de processamento gastos para codificar/decodificar cada uma das imagens da Figura 7.3.

Tabela 7.1 – Resultados obtidos do codificador de Fisher.

Imagem	*Taxa de bits (bpp)*	*Tempo (s)*	*PSNR (dB)*
Lena	0,20	915	28,43
	0,25	1370	29,13
	0,35	2175	30,58
	0,45	3085	31,54
	0,49	3460	32,33
Couple	0,20	975	25,00
	0,25	1410	25,40
	0,35	2144	26,50
	0,45	3042	27,70
	0,49	3440	28,40
Swan	0,20	946	23,00
	0,25	1392	23,40
	0,35	2192	24,10
	0,45	3105	25,50
	0,49	3522	26,50

7.4.3- Resultados Obtidos do Quantizador Vetorial

As Figuras 7.7 a 7.9 mostram os resultados obtidos ao utilizar o quantizador vetorial, a uma taxa de compressão de aproximadamente 0,49 bpp.

(a)

(b)

Figura 7.7 – Resultado obtido da codificação da imagem Lena pelo quantizador vetorial à 0,49 bpp: (a) imagem original; (b) imagem reconstruída – PSNR = 30,63 dB.

(a)

(b)

Figura 7.8 – Resultado obtido da codificação da imagem Couple pelo quantizador vetorial à 0,49 bpp: (a) imagem original; (b) imagem reconstruída – PSNR = 27,81 dB.

(a)

(b)

Figura 7.9 – Resultado obtido da codificação da imagem Swan pelo quantizador vetorial à 0,49 bpp: (a) imagem original; (b) imagem reconstruída – PSNR = 26,19 dB.

A Tabela 7.2 mostra os resultados obtidos ao utilizar o quantizador vetorial, bem como os tempos de processamento aproximados para codificação/decodificação de cada uma das imagens.

Tabela 7.2 – Resultados obtidos do quantizador vetorial.

Imagem	*Taxa de bits (bpp)*	*Tempo (s)*	*PSNR (dB)*
Lena	0,20	5	27,22
	0,25	5	28,15
	0,35	6	28,9
	0,45	7	30,29
	0,49	7	30,63
Couple	0,20	4	22,98
	0,25	5	24,08
	0,35	6	25,10
	0,45	6	26,50
	0,49	7	27,80
Swan	0,20	5	22,00
	0,25	5	22,32
	0,35	6	23,22
	0,45	6	24,50
	0,49	7	26,19

7.4.4- Resultados Obtidos do Codificador Fractal-VQ Proposto neste Trabalho

As Figuras 7.10 a 7.14 mostram os resultados obtidos ao utilizar o codificador Fractal-VQ proposto neste trabalho. As imagens Couple e Swan foram codificadas à 0.49 e 0.72 bpp e a imagem Lena foi codificada à taxa de 0.49 bpp.

(a)

(b)

Figura 7.10 – Resultado obtido da codificação da imagem Lena pelo codificador Fractal-VQ proposto à 0,49 bpp: (a) imagem original; (b) imagem reconstruída – PSNR = 33,89 dB.

(a)

(b)

Figura 7.11 – Resultado obtido da codificação da imagem Couple pelo codificador Fractal-VQ proposto à 0,49 bpp: (a) imagem original; (b) imagem reconstruída – PSNR = 29,34 dB.

(a)

(b)

Figura 7.12 – Resultado obtido da codificação da imagem Couple pelo codificador Fractal-VQ proposto à 0,72 bpp: (a) imagem original; (b) imagem reconstruída – PSNR = 31,05 dB.

(a)

(b)

Figura 7.13 – Resultado obtido da codificação da imagem Swan pelo codificador Fractal-VQ proposto à 0,49 bpp: (a) imagem original; (b) imagem reconstruída – PSNR = 27,40 dB.

(a)

(b)

Figura 7.14 – Resultado obtido da codificação da imagem Swan pelo codificador Fractal-VQ proposto à 0,72 bpp: (a) imagem original; (b) imagem reconstruída – PSNR = 29,27 dB.

Para uma análise mais objetiva, a Tabela 7.3 apresenta os resultados obtidos ao utilizar o codificador Fractal-VQ proposto neste trabalho.

Tabela 7.3 – Resultados obtidos do codificador Fractal-VQ.

Imagem	*Taxa de bits (bpp)*	*Tempo (s)*	*PSNR (dB)*
Lena	0,20	154	29,76
	0,25	163	30,44
	0,35	178	31,97
	0,45	183	33,38
	0,49	198	33,89
Couple	0,20	157	25,90
	0,25	170	26,59
	0,35	186	27,86
	0,45	192	28,86
	0,49	210	29,34
	0,72	269	31,05
Swan	0,20	170	24,50
	0,25	179	24,88
	0,35	184	25,87
	0,45	194	27,06
	0,49	217	27,40
	0,72	263	29,27

7.4.5- Análise dos Resultados

Comparando-se subjetivamente as imagens reconstruídas dos codificadores, pode-se verificar que todas elas apresentam uma boa qualidade visual, considerando-se as taxas de bits correspondentes. Pode-se observar que o codificador Fractal-VQ atingiu valores de PSNR superiores aos obtidos pelo quantizador vetorial e pelo codificador de Fisher, podendo-se concluir que o codificador proposto, em termos de fidelidade, para as taxas de bits avaliadas, proporciona resultados superiores aos outros. Os gráficos das Figuras 7.15 a 7.17 mostram claramente a superioridade do codificador proposto referente à PSNR na codificação das imagens ilustradas na Figura 7.3, à variadas taxas de bits.

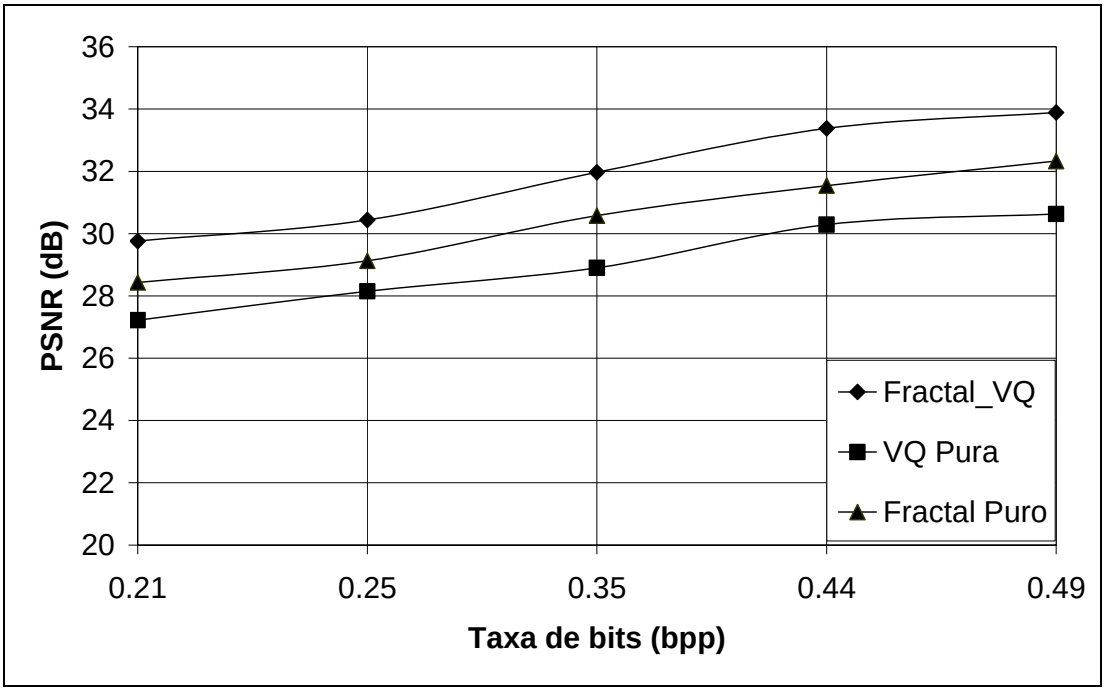


Figura 7.15 – Gráfico comparativo da taxa bits versus PSNR para a imagem Lena entre o codificador Fractal-VQ, quantizador vetorial e codificador fractal puro.

Pode-se observar que para as imagens que apresentam mais riqueza de detalhes, tais como texturas, tonalidades e contornos, o quantizador vetorial, à 0,49 bpp, aproxima-se do codificador de Fisher para este tipo de imagem, conforme análise dos resultados da imagem Couple e Swan nos gráficos das Figuras 7.16 e 7.17. Dessa forma, pode-se concluir que para imagens que apresentam pouca auto-similaridade, a codificação fractal pura não se sobressai, tornando-se uma alternativa cara do ponto vista custo-benefício.

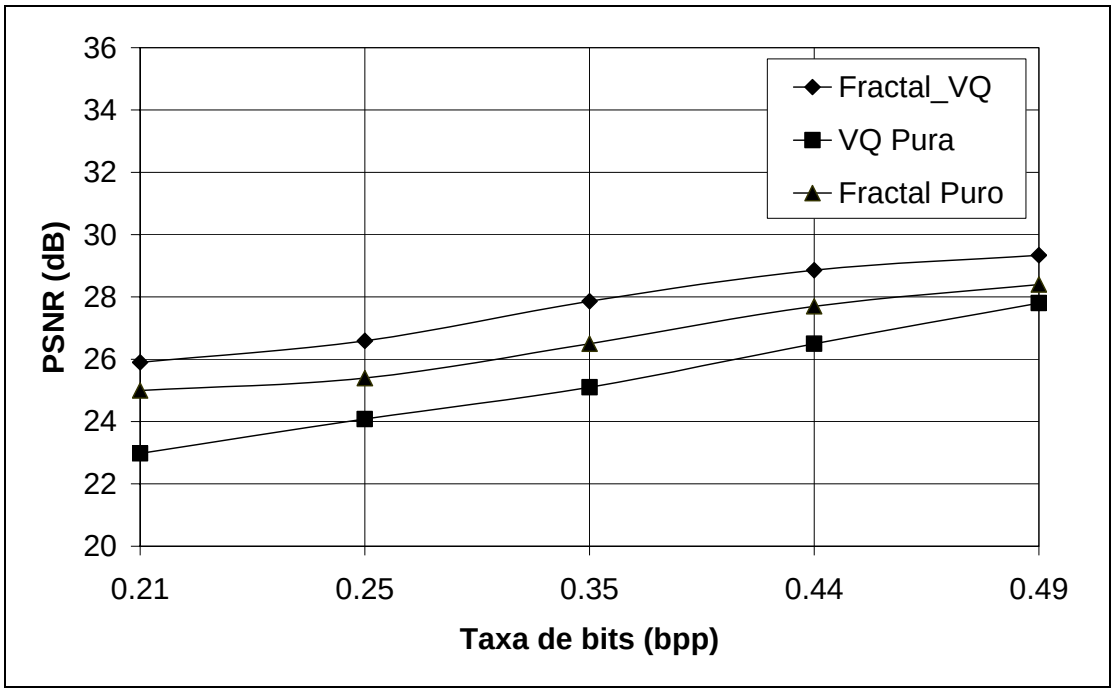


Figura 7.16 – Gráfico comparativo da taxa bits versus PSNR para a imagem Couple entre o codificador Fractal-VQ, quantizador vetorial e codificador fractal puro.

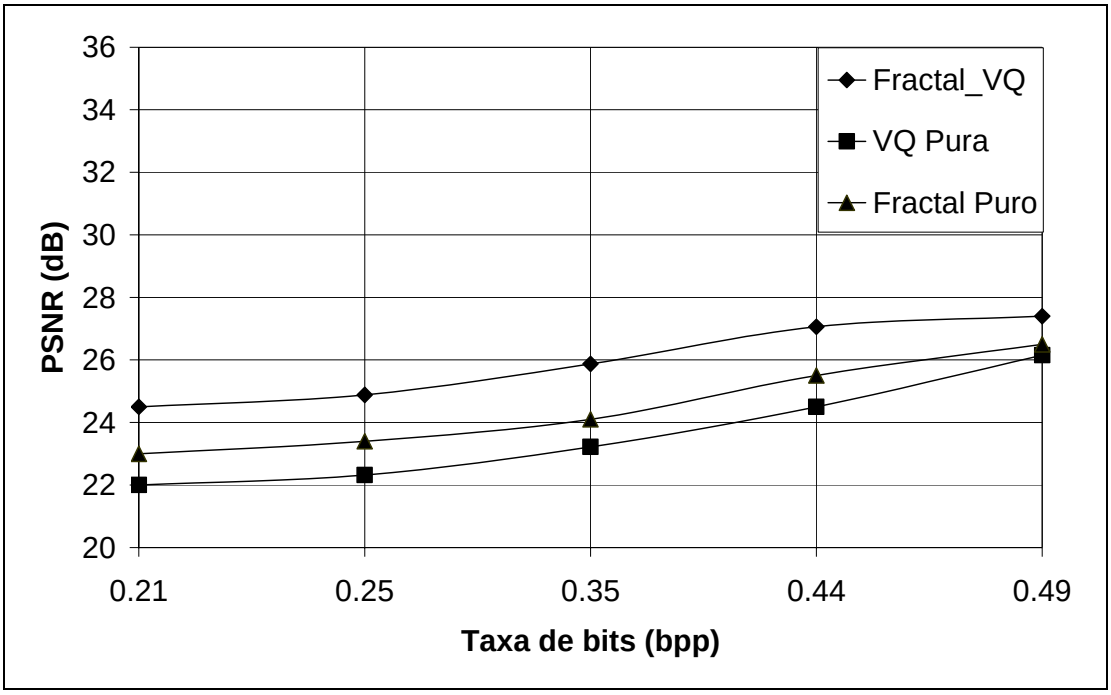


Figura 7.17 – Gráfico comparativo da taxa bits versus PSNR para a imagem Swan entre o codificador Fractal-VQ, quantizador vetorial e codificador fractal puro.

Na maioria dos sistemas codificadores de imagens com perdas, a altas taxas de compressão, existe a presença do artefato conhecido como blocagem, que é uma descontinuidade nos contornos dos blocos da imagem reconstruída. Na codificação fractal esse efeito é minimizado, em relação à quantização vetorial, em virtude da exploração da auto-similaridade presente nas imagens digitais e flexibilidade do ajuste de brilho e contraste, que são as características mais importantes da codificação fractal. No modelo proposto este artefato é menos visível que na codificação fractal pura, sendo que à taxas de bits mais elevadas tal artefato é minimizado consideravelmente, conforme pode-se observar nas Figuras 7.12 e 7.14 quando comparadas às Figuras 7.11 e 7.13, respectivamente.

O quantizador vetorial apresenta o menor tempo de processamento utilizado para codificar e decodificar as imagens utilizadas nos testes, sendo que na codificação, para cada bloco a ser processado, basta pesquisar no *codebook* e transmitir o índice do vetor com menor

distância. Já a decodificação envolve apenas operações de indexação, isto é, recuperar os vetores (blocos da imagem) com uma simples busca no *codebook* dos índices transmitidos. O codificador Fractal-VQ apresenta menor tempo de processamento em relação ao codificador de Fisher devido a dois fatores: 1) o *domain-pool* é um *codebook* genérico, não sendo necessário designar um *domain-pool* para cada imagem a ser codificada, composto por um número bastante reduzido de vetores, minimizando substancialmente o número de operações necessárias para se encontrar o melhor vetor candidato; e 2) a decodificação é uma operação de indexação, semelhante ao da quantização vetorial, onde o *codebook* também pode ser visto como a imagem inicial do processo de decodificação, porém, apenas uma iteração é suficiente para a recuperação da imagem codificada. Os gráficos das Figuras 7.18 a 7.20 mostram as curvas referentes aos tempos de processamento do codificador fractal puro e do codificador Fractal-VQ para as três imagens estudadas, a variadas taxas de bits.

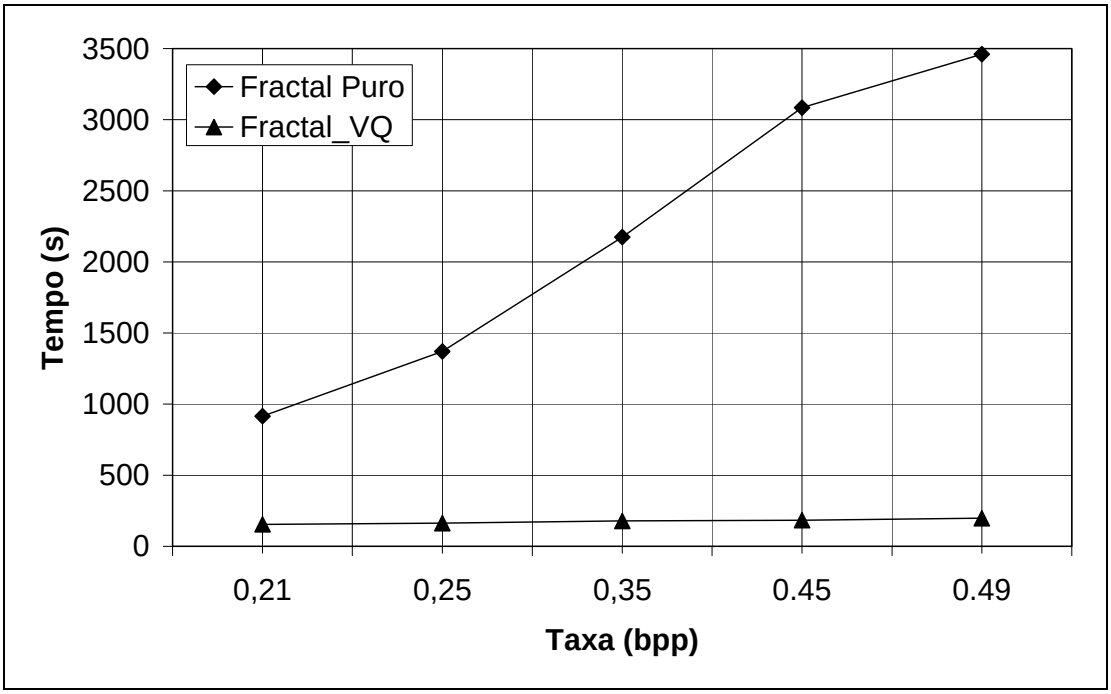


Figura 7.18 – Gráfico comparativo do tempo de processamento entre o codificador de Fisher e o codificador Fractal-VQ na codificação da imagem Lena.

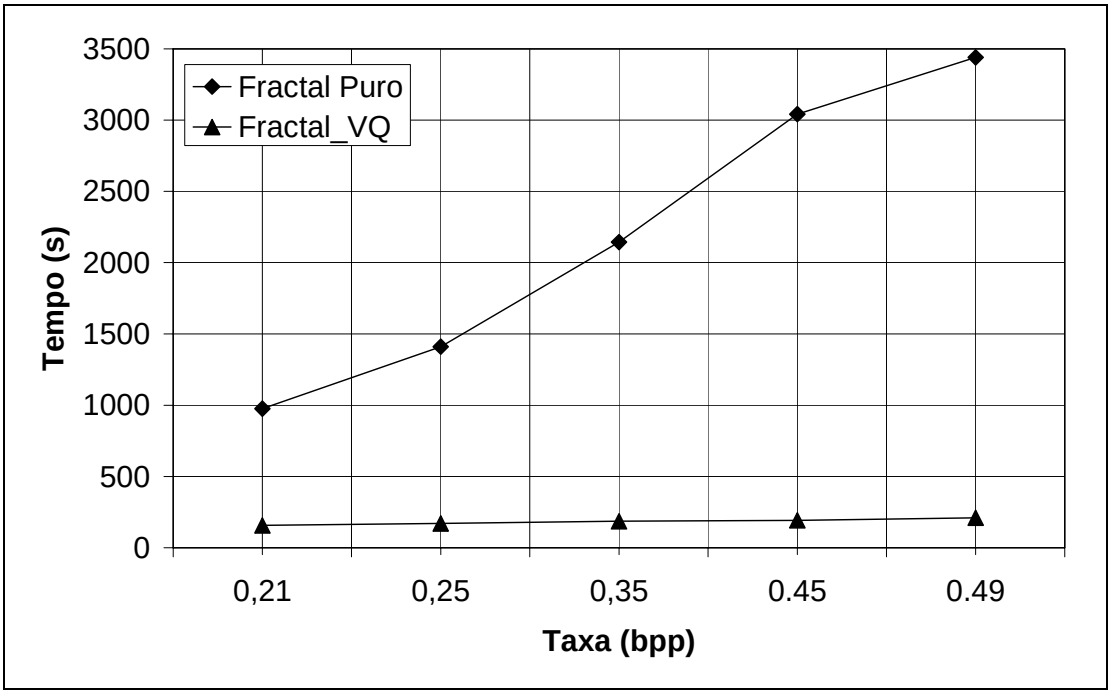


Figura 7.19 – Gráfico comparativo do tempo de processamento entre o codificador de Fisher e o codificador Fractal-VQ na codificação da imagem Couple.

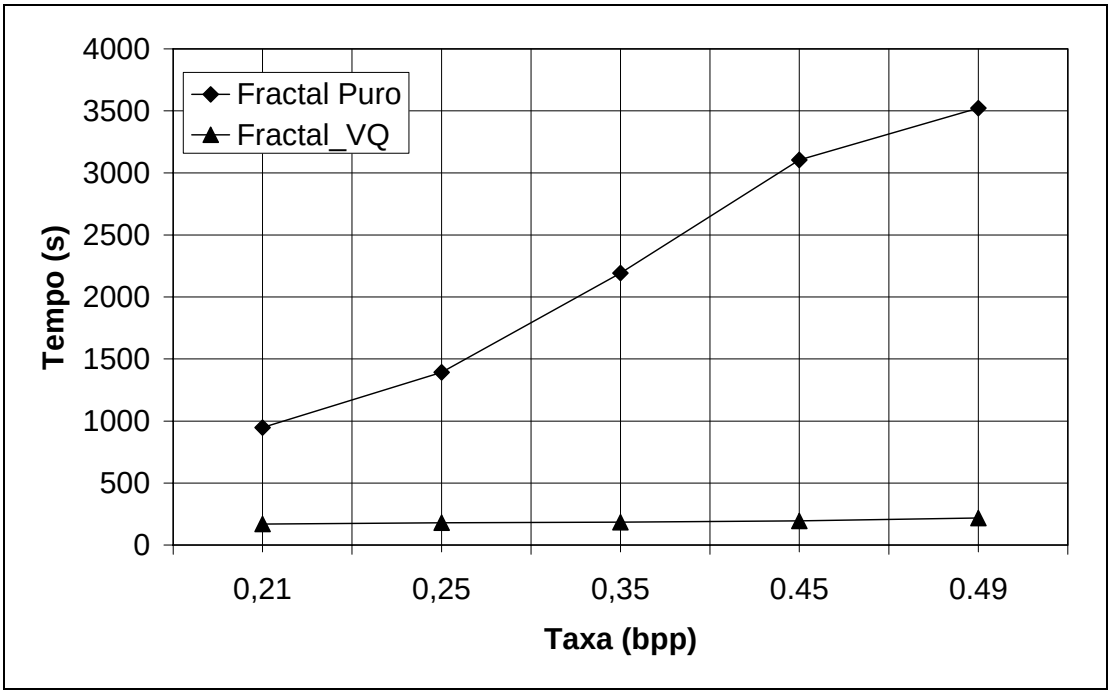


Figura 7.20 – Gráfico comparativo do tempo de processamento entre o codificador de Fisher e o codificador Fractal-VQ na codificação da imagem Swan.

7.4.6- Comparação entre o Codificador Fractal-VQ Proposto neste Trabalho e Um Codificador Fractal-Wavelet

Nesta seção são apresentados os resultados e comparações entre o codificador Fractal-VQ proposto neste trabalho e o codificador Fractal-Wavelet apresentado por Iano, Silva e Cruz [12], que é uma referência bastante atualizada e cujos resultados se mostram eficazes, principalmente em termos da velocidade de processamento, tornando tal modelo uma boa referência de comparação. Vale ressaltar que esse codificador não foi implementado neste trabalho, de forma que os resultados obtidos do codificador Fractal-Wavelet apresentados a seguir foram extraídos de Iano, Silva e Cruz [12]. As imagens utilizadas nas simulações para o codificador Fractal-VQ proposto neste trabalho foram as mesmas usadas por Iano, Silva e Cruz [12], ilustradas na Figura 7.21.

(a) (b) (c)

Figura 7.21 – Imagens 512 x 512 pixels e 8 bpp; (a) Lena; (b) Goldhill; (c) Boat.

A imagem Lena codificada pelo modelo Fractal-VQ, a 0,49 bpp, pode ser vista na Figura 7.10 e os resultados obtidos à variadas taxas de bits podem ser conferidos na Tabela 7.3 deste capítulo. As Figuras 7.22 e 7.23 apresentam as imagens Goldhill e Boat, respectivamente, codificadas pelo modelo Fractal-VQ à 0,72 bpp, ou seja, alcançando uma taxa de compressão de aproximadamente 11:1.

(a)

(b)

Figura 7.22 – Resultado obtido da codificação da imagem Goldhill pelo codificador Fractal-VQ proposto à 0,72 bpp: (a) imagem original; (b) imagem reconstruída – PSNR = 32,50 dB.

(a)

(b)

Figura 7.23 – Resultado obtido da codificação da imagem Boat pelo codificador Fractal-VQ proposto à 0,72 bpp: (a) imagem original; (b) imagem reconstruída – PSNR = 33,14 dB.

Os gráficos das Figuras 7.24 a 7.26 mostram as curvas PSNR do codificador Fractal-VQ proposto, codificador Fractal-Wavelet e codificador fractal puro (Fisher) para as imagens da Figura 7.21, à variadas taxas de bits, lembrando que os dados referentes ao codificador Fractal-Wavelet e ao codificador fractal puro, nessa seção, foram extraídos de Iano, Silva e Cruz [12].

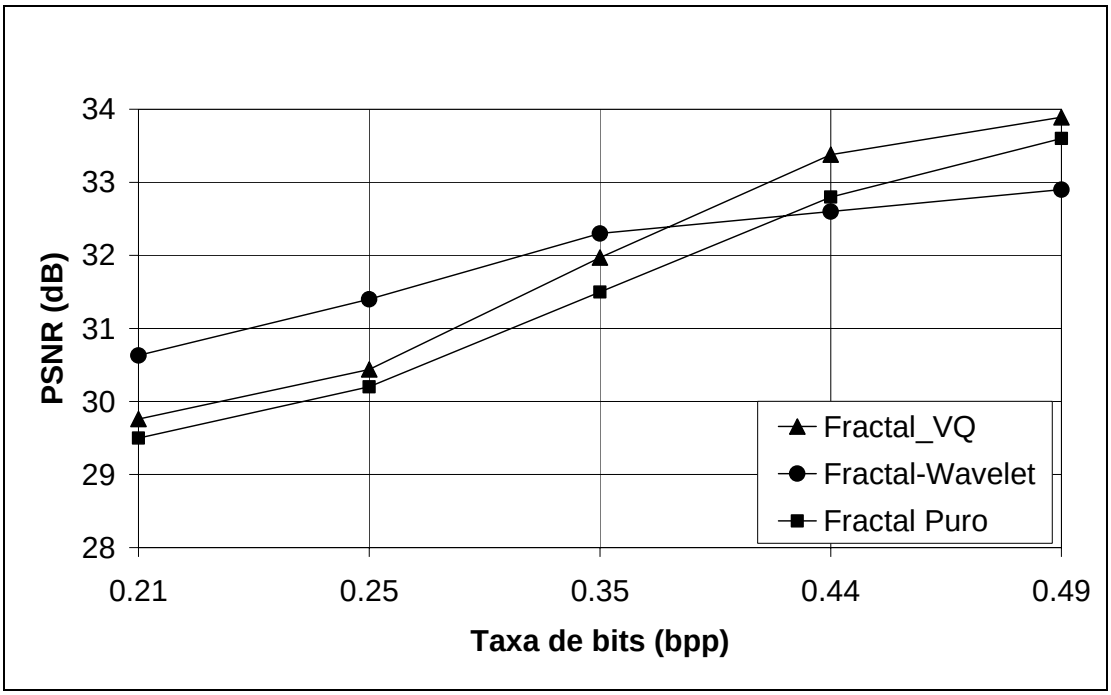


Figura 7.24 – Gráfico comparativo da taxa bits versus PSNR para a imagem Lena entre o codificador Fractal-VQ, codificador Fractal-Wavelet e o codificador fractal puro.

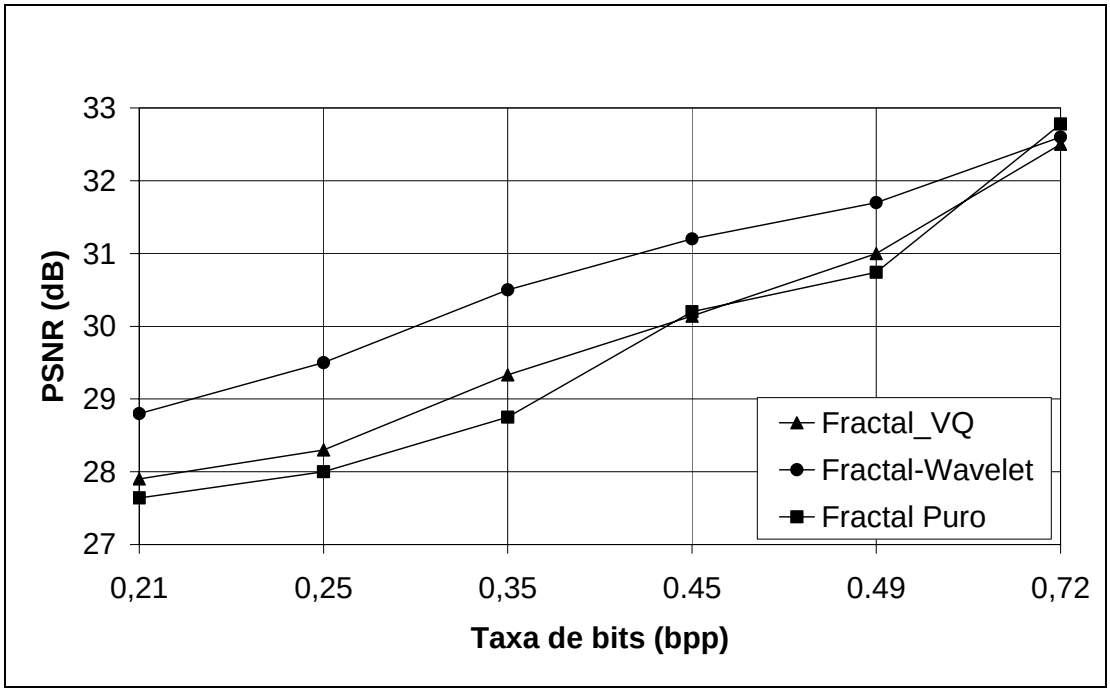


Figura 7.25 – Gráfico comparativo da taxa bits versus PSNR para a imagem Goldhill entre o codificador Fractal-VQ, codificador Fractal-Wavelet e o codificador fractal puro.

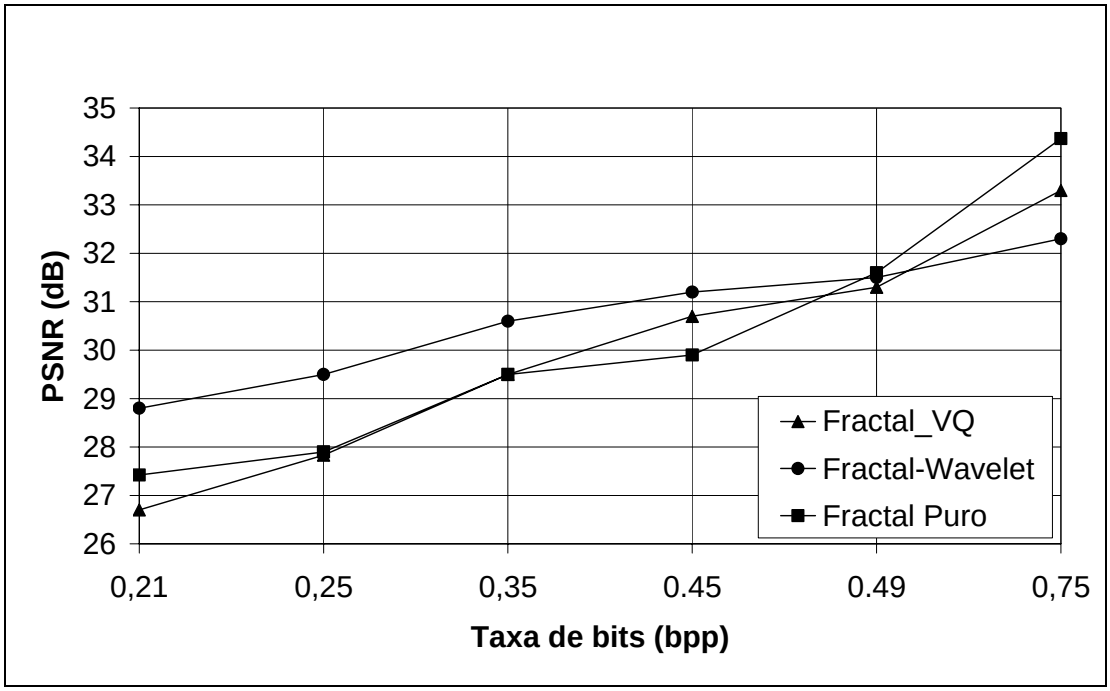


Figura 7.26 – Gráfico comparativo da taxa bits versus PSNR para a imagem Boat entre o codificador Fractal-VQ, codificador Fractal-Wavelet e o codificador fractal puro.

Pode-se observar, nos gráficos das Figuras 7.24 a 7.26, que para a imagem Lena, o codificador Fractal-VQ atinge melhores valores de PSNR que o codificador fractal puro, superando o codificador Fractal-Wavelet quando a taxa de bits é superior a 0,35 bpp. Para a imagem Goldhill, o codificador proposto supera ligeiramente o codificador fractal puro na maioria dos casos e atinge valores de PSNR muito próximos aos do codificador Fractal-Wavelet à 0,72 bpp, tendendo a superá-lo para taxas de bits superiores. Para a imagem Couple, os três codificadores à 0,49 bpp atingem valores de PSNR muito próximos, onde o codificador Fractal-VQ é superior ao codificador Fractal-Wavelet a partir desse ponto.

Comparando-se o tempo de processamento utilizado para a codificação/decodificação das imagens Lena, Goldhill e Boat, entre o codificador Fractal-VQ proposto e o codificador Fractal-Wavelet, pode-se constatar a superioridade do modelo Fractal-VQ proposto neste trabalho, como pode ser observado nos gráficos das Figuras 7.27 a 7.29.

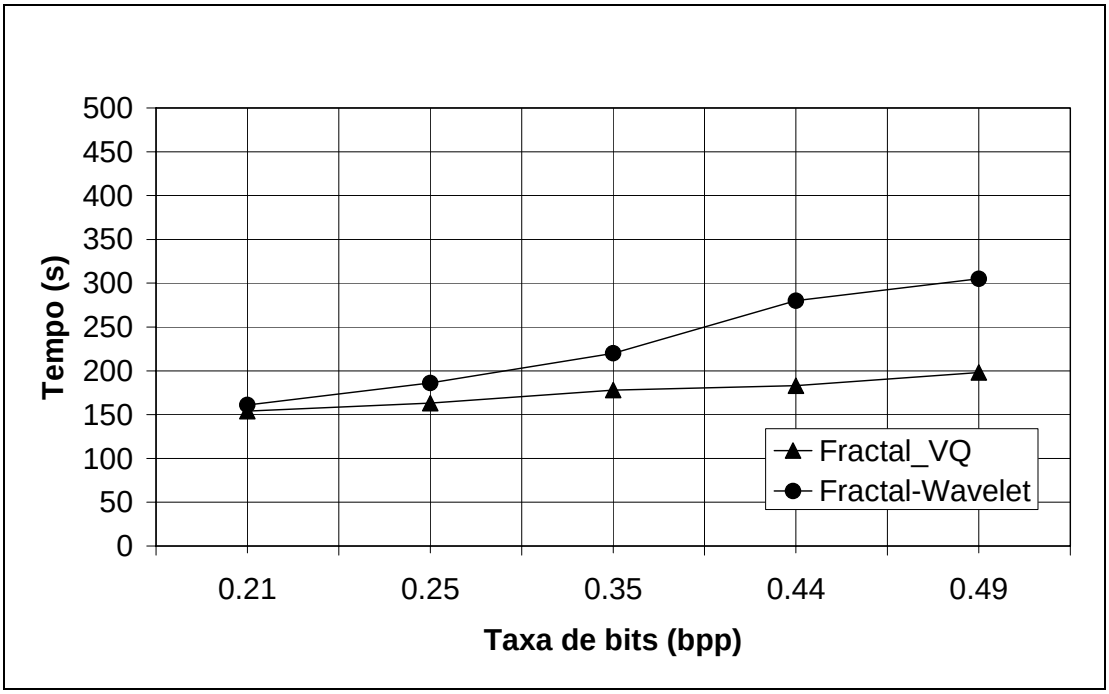


Figura 7.27 – Gráfico comparativo da taxa bits versus tempo de processamento para a imagem Lena entre o codificador Fractal-VQ e o codificador Fractal-Wavelet.

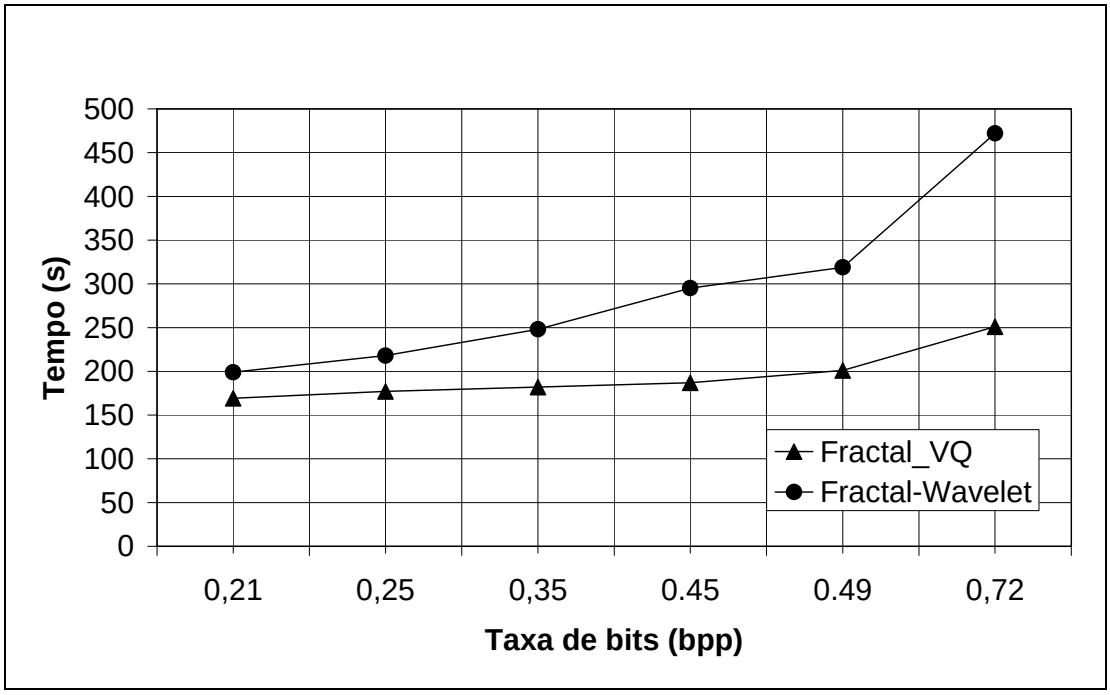


Figura 7.28 – Gráfico comparativo da taxa bits versus tempo de processamento para a imagem Goldhill entre o codificador Fractal-VQ e o codificador Fractal-Wavelet.

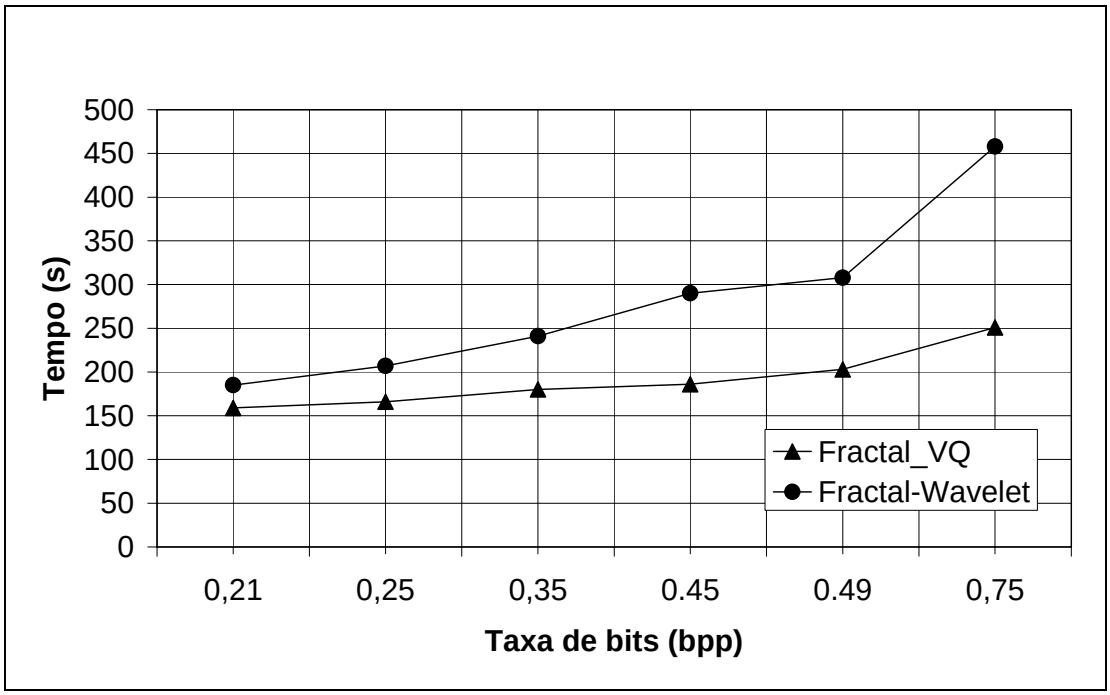


Figura 7.29 – Gráfico comparativo da taxa bits versus tempo de processamento para a imagem Boat entre o codificador Fractal-VQ e o codificador Fractal-Wavelet.

A Tabela 7.4 mostra um resumo dos valores apresentados nos gráficos das Figuras 7.24 a 7.29.

Tabela 7.4 – Comparação dos resultados obtidos entre o codificador Fractal-VQ e o codificador Fractal-Wavelet.

Imagem	*Taxa de bits (bpp)*	*Modelo de Codificação*	*Tempo (s)*	*PSNR (dB)*
Lena	0,21	Fractal-VQ	154	29,76
		Fractal-Wavelet	161	30,63
	0,25	Fractal-VQ	163	30,44
		Fractal-Wavelet	186	31,40
	0,35	Fractal-VQ	178	31,97
		Fractal-Wavelet	220	32,30
	0,45	Fractal-VQ	183	33,38
		Fractal-Wavelet	280	32,60
	0,49	Fractal-VQ	198	33,89
		Fractal-Wavelet	305	32,90
Goldhill	0,21	Fractal-VQ	169	27,90
		Fractal-Wavelet	199	28,80
	0,25	Fractal-VQ	177	28,30
		Fractal-Wavelet	218	29,50
	0,35	Fractal-VQ	182	29,33
		Fractal-Wavelet	248	30,50
	0,45	Fractal-VQ	187	30,14
		Fractal-Wavelet	295	31,20
	0,49	Fractal-VQ	201	31,00
		Fractal-Wavelet	319	31,70
	0,72	Fractal-VQ	251	32,50
		Fractal-Wavelet	472	32,60

Boat	0,21	Fractal-VQ	159	26,70
		Fractal-Wavelet	185	28,80
	0,25	Fractal-VQ	166	27,83
		Fractal-Wavelet	207	29,50
	0,35	Fractal-VQ	180	29,50
		Fractal-Wavelet	241	30,60
	0,45	Fractal-VQ	186	30,70
		Fractal-Wavelet	290	31,20
	0,49	Fractal-VQ	203	31,30
		Fractal-Wavelet	308	31,50
	0,75	Fractal-VQ	251	33,30
		Fractal-Wavelet	458	32,30

7.5- Conclusão

Neste capítulo foram apresentados os resultados e comparações envolvendo o codificador de imagens proposto neste trabalho e outros modelos de codificação referenciados, mostrando que o esquema proposto produz bons resultados, tanto em termos de PSNR como em desempenho. Muitas melhorias ainda podem ser implementadas fazendo com que o codificador Fractal-VQ proposto nesta tese fique ainda mais rápido, mantendo alta fidelidade da imagem reconstruída em relação à original.

No próximo capítulo serão mostradas as conclusões gerais da tese, principais contribuições e sugestões para futuros trabalhos.

CAPÍTULO VIII

CONCLUSÕES GERAIS, CONTRIBUIÇÕES E PROPOSTAS PARA TRABALHOS FUTUROS

8.1- Conclusões Gerais

A compressão de imagens digitais tornou-se uma das principais áreas de pesquisas no ramo de processamento digital de imagens (PDI). Atualmente, uma enorme variedade de modelos de codificação vem sendo apresentada, entretanto, pode-se notar que as principais técnicas descritas no Capítulo II deste trabalho, tanto de compressão com perdas como compressão sem perdas, permanecem em evidência há vários anos, principalmente pelo fato de que cada uma delas apresenta características peculiares que as tornam boas alternativas para os diversos tipos de aplicações existentes nessa área. Dessa forma, o grande objetivo das pesquisas em compressão de imagens envolve a construção de sistemas de codificação híbridos que aglomeram as melhores características de duas ou mais técnicas procurando-se criar modelos cada vez mais eficientes.

A codificação de imagens por fractais é relativamente recente, ao contrário da geometria clássica, cujos conceitos são muito antigos, os primeiros estudos relacionados à geometria fractal aconteceram em 1975 e o primeiro codificador de imagens foi apresentado por Jacquin [4] no início da década de 90, esse codificador foi o marco do início de uma série de pesquisas relacionadas ao assunto. Logo em seguida, Fisher [7] desenvolveu um

codificador fractal considerado muito eficiente, que é utilizado como base de comparação até hoje. Atualmente, por já estar bem estabelecida, a codificação fractal é muito utilizada em conjunto com outras técnicas para produzir novos modelos de compressão.

A quantização vetorial é uma técnica bastante conhecida e muito utilizada em processamento digital de sinais (PDS), principalmente na codificação de voz e imagem. Apesar de existirem vários métodos para a geração de *codebooks*, o algoritmo LBG, apresentado por Linde, Buzo e Gray [14], é um dos mais utilizados atualmente e por ser considerado muito eficiente foi utilizado neste trabalho.

De acordo com os resultados experimentais obtidos no Capítulo VII, o modelo de codificação Fractal-VQ proposto neste trabalho, descrito no Capítulo VI, apresentou resultados satisfatórios, mantendo bons níveis de PSNR, com a grande vantagem de proporcionar uma diminuição de aproximadamente 85% a 95% , conforme a taxa de bits utilizada, do tempo de codificação/decodificação das imagens estudadas, quando comparado ao codificador fractal puro (codificador de Fisher).

Quando comparado a outro codificador híbrido Fractal-Wavelet [12], o codificador Fractal-VQ mostrou-se equiparado em termos de PSNR com uma aparente melhora de desempenho, exigindo menor esforço computacional para codificar/decodificar as imagens digitais da Figura 7.21, conforme pode ser observdo na Seção 7.4.6 do Capítulo VII deste trabalho.

8.2- Contribuições deste Trabalho

Nesta tese destacam-se como principais contribuições:

- Descrição do estado da arte atual referente às técnicas de compressão de imagens digitais;
- Exposição da fundamentação teórica sobre a geometria fractal e a codificação de imagens por fractais, bem como apresentação dos resultados, obtidos na implementação dos algoritmos de um codificador fractal de imagens digitais.
- Detalhamento do funcionamento da técnica de quantização vetorial (VQ) e apresentação dos resultados obtidos na implementação de um quantizador vetorial de imagens digitais; e
- Desenvolvimento de um novo codificador híbrido de imagens combinando a codificação fractal e a quantização vetorial (VQ), bem como a apresentação dos resultados obtidos e a realização de comparações entre esse codificador e outros codificadores presentes na literatura. A utilização de um *codebook* genérico bastante reduzido e o uso da DCT para a escolha da transformação geométrica (isometria) mais adequada, quando comparado a outros codificadores estudados, reduz substancialmente o esforço computacional necessário para a codificação/decodificação de imagens digitais, mantendo bons níveis de PSNR.

8.3- Proposta para Trabalhos Futuros

A codificação de imagens é um assunto que apresenta ampla aplicabilidade e, apesar de encontrar-se em avançado estágio de exploração, muito ainda pode ser investigado. Uma diversidade de modelos de compressão pode ser desenvolvida, combinando-se duas ou mais

das variadas técnicas descritas no Capítulo II deste trabalho, levando-se em consideração o tipo de aplicação envolvida.

Acredita-se que o codificador Fractal-VQ proposto neste trabalho pode apresentar resultados ainda melhores pela implementação de algumas sugestões, dentre as quais se destacam:

- Adotar outros tipos de partições adaptativas, como as ilustradas na Figura 4.2 deste trabalho, onde a triangulação de Delaunay pode ser uma boa alternativa para minimizar o efeito de blocagem, artefato muito comum encontrado nas imagens reconstruídas produzido pela maioria dos codificadores de imagens com perdas. Maiores detalhes sobre partições adaptativas, incluindo a triangulação de Delaunay podem ser encontrados em [8, 33, 34 e 35];
- Explorar técnicas de classificação de blocos mais sofisticadas, buscando diminuir cada vez mais o tempo gasto na codificação mantendo a qualidade da imagem reconstruída. Alguns métodos de classificação podem ser encontrados em [27, 36, 40, 41, 42, 43 e 60]; e
- Buscar outros mecanismos para se encontrar a transformação geométrica (isometria) mais adequada. Neste trabalho foi utilizado um método baseado no produto interno da DCT, descrito na seção 6.2.1 do Capítulo VI.

8.4- Conclusão

A proposta desta pesquisa foi desenvolver um modelo híbrido de codificação de imagens digitais combinando as técnicas de codificação fractal e quantização vetorial. Os objetivos propostos no Capítulo I foram alcançados, resultando nas contribuições descritas na

Seção 8.2 deste capítulo. Também foram apresentadas sugestões para futuros trabalhos propondo melhorias a serem implementadas no codificador Fractal-VQ.

Espera-se que este trabalho tenha contribuído para o enriquecimento da literatura no que diz respeito à compressão de imagens digitais, sabendo-se que tal área atrai muitas pesquisas e continuará evoluindo cada vez mais.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Mandelbrot, B. B.. *The Fractal Geometry of Nature*, W. H. Freeman, New York, 1982, 468 p.

[2] Barnsley, M. F.. *Fractals Everywhere*, Academic Press, Boston,1988, 394 p.

[3] Barnsley, M., Hurd, L.. *Fractal Image Compression*, AK Peters, Wellesley, 1993.

[4] Jacquin, A. E., *Image coding based on a Fractal Theory of Iterated Contractive Image Transformations*, IEEE trans. on Image Processing, pp. 18-30, Jan 1992.

[5] Jacquin, A. E., *Fractal Image Coding : A Review*, Proceedings of the IEEE, vol.81, no.10, Oct. 1993.

[6] Fisher, Y., *Fractal Image Compression*, SIGGRAPH 92 course notes, 1992.

[7] Fisher, Y.. *Fractal Image Compression: Theory and Application*, New York: Springer-Verlag, 1994.

[8] Davoine, F., Antonini, M., ChasserY, J., Barlaud, M.. *Fractal Image Compression Based on Delaunay Triangulation and Vector Quantization*, IEEE Transactions on Image Processing, vol. 5, no. 2, pp. 338 – 346, Feb 1996.

[9] Thao, T. N.. *A Hybrid Fractal-DCT Coding Scheme for Image Compression*, IEEE Proceedings - International Conference on Image Processing - ICIP-96, vol. 1, pp. 169-172, Sep. 1996.

[10] Kim, I. K., Park, R.. *Still Image Coding Based on Vector Quantization and Fractal Approximation*, IEEE Transactions on Image Processing, vol. 5, no. 4, pp. 587 – 597, Apr. 1996.

[11] Hamzaoui, R., Saupe, D.. *Combining Fractal Image Compression and Vector Quantization*, IEEE Transactions on Image Processing, vol. 9, no. 2, pp. 197 – 208, Feb. 2000.

[12] Iano, Y., da Silva, F. S., Cruz, A. L. M.. *A Fast and Efficient Hybrid Fractal-Wavelet Image Coder*, IEEE Transactions on Image Processing, vol. 15, no. 1, pp. 98 – 105, Jan. 2006.

[13] Gray, R. M.. *Vector quantization*, IEEE ASSP Magazine, vol. 1, no. 2, pp. 4-29, Apr. 1984.

[14] Linde, Y., Buzo, A., Gray, R. M.. *An Algorithm for Vector Quantizer Design*, IEEE Transactions Communications, vol. 28, no.1, pp. 84-95, Jan. 1980.

[15] Vasuki, A., Vanathi, P. T.. *A Review of Vector Quantization Techniques*, IEEE Potentials, vol. 25, no. 4, pp. 39-47, Jul./Aug. 2006.

[16] Gonzales, R. C., Woods, R. E. *Processamento de Imagens Digitais*. Tradução por Roberto Marcondes César Junior e Luciano da Fontoura Costa. São Paulo: Editora Edgard Blücher Ltda, 2000.

[17] Subramanya, S. R.. *Image Compression Techniques*. IEEE Potentials, vol. 20, no. 1, pp. 19-23, Feb/Mar. 2001.

[18] Yang, M., Bourbakis, N.. *An Overview of Lossless Digital Image Compression Techniques*. IEEE Proceedings - 48th Midwest Symposium on Circuits and Systems, pp. 1099-1102, Aug. 2005.

[19] Jaisimha, M. Y., Potlapalli, H., Barad, H., Martinez, A. B., Lohrenz, M. C., Ryan, J., Pollard, J.. *Data Compression Techniques for Maps*. IEEE Proceedings Southeastcon, vol.2, pp. 878–883, Apr. 1989.

[20] Arps, R. B., Truong, T. K.. *Comparison of International Standards for Lossless Still Image Compression*, Proceedings of the IEEE, vol. 82, no. 6, pp. 889-899, Jun. 1994.

[21] Jain, A. K.. *Fundamentals of Digital Image Processing*. Englewood Cliffs, NJ: Prentice Hall, 1989.

[22] Fränti, P., Nevalainen, O.. *Block Truncation Coding with Entropy Coding*. IEEE Trans. Commun., vol. 43, no. 2/3/4, pp. 1677-1685, Feb./Mar./Apr. 1995.

[23] Wu, Y. –G., Chuan, S.. *Medical Image Compression by Discrete Cosine Transform Similarity Strategy*. IEEE Trans. Inf. Technol. Biomed., vol. 5, no. 3, pp. 236-243, Sep. 2001.

[24] Jayant, N. S., Noll, P.. *Digital Coding of Waveforms*. Englewood Cliffs, NJ: Prentice Hall, 1984.

[25] Gharavi, H., Tabatabai, A.. *Sub-band Coding of Monochrome and Color Images*. IEEE Trans. Circ. and Syst., vol. 35, no. 2, pp. 207-214, Feb. 1988.

[26] Tan, R. H. G., Zhang, J. F., Morgan, R., Greenwood, A.. *Still Image Compression Based on 2D Discrete Wavelet Transform*. IEEE Electronic Letters Online, vol. 35, no. 22, pp. 1934-1935, Oct. 1999.

[27] Wohlberg, B., Jager, G.. *A Review of the Fractal Image Coding Literature*. IEEE Trans. Imag. Proc., vol. 8, no. 12, pp. 1716-1729, Dec. 1999.

[28] Wallace, G.. *The JPEG Still Picture Compression Standard*. Commun. ACM, vol. 34, pp. 30-44, Apr. 1991.

[29] Douglas, C., Bouzerdoum, A.. *JPEG2000 Image Compression: An Overview*, IEEE Proceedings - Seventh Australian and New Zealand Intelligent Information System Conference, pp. 237-241, Nov. 2001.

[30] Bojkovicé, Z., Milovanovié, D.. *Advanced Image Compression Techniques: Comparative Functionalities and Performances*, IEEE Proceedings - 5th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Service - TELSIKS 2001, pp. 439-444, Sep. 2001.

[31] Falconer, K. *Fractal Geometry*, 2nd ed., Wiley, New York, 2003.

[32] Hart, John C., *Fractal Image Compression and Recurrent Iterated Function Systems*, IEEE Computer Graphics and Applications, vol. 16, no. 4, pp. 25-33, Jul 1996.

[33] Davoine, F., Chassery, J.-M.. *Adaptative Delaunay Triangulation for Attractor Image Coding*, 12th International Conference on Pattern Recognition, IEEE Oct. 1994.

[34] Ponomarenko, M., Egiazarian, K., Lukin, V., Astola, J.. *Compression of Image Block Means for Non-equal Block Partition Schemes using Delaunay Triangulation and Prediction*, IEE Proceedings - Vision, Image and Signal Processing , vol. 150, no 4, pp. 239-243, Aug. 2003.

[35] Ruhl, M., Hartenstein, H., Saupe, D.. *Adaptative Partitionings for Fractal Image Compression*, IEEE Proceedings - International Conference on Image Processing, pp. 310-313, Oct. 1997.

[36] Saupe, D., Hamzaoui, R.. *Complexity Reduction Methods for Fractal Image Compression*, Proc. IMA Conf. Proc. on Image Processing; Mathematical Methods and Applications, Sept. 94, J. M. Blackledge (ed.), Oxford University Press, 1997.

[37] Ramamurthi, B., Gersho, A.. *Classified Vector Quantization of Images*, IEEE Trans. Commun., vol. 34, Nov. 1986.

[38] Jacobs, E.W., Ross, R.D., Fisher, Y.. *Fractal-based Image Compression II*, Technical Report 1362, Naval Ocean Systems Center, San Diego, 1990.

[39] Yung-Gi, W., Ming-Zhi, H., Yu-Ling, W.. *Fractal Image Compression with variance and Mean*, IEEE Proceedings - International Conference on Multimedia and Expo - IEEE ICME 2003, vol. 1, pp. 353 – 356, Jul 2003.

[40] Saupe, D.. *Accelerating Fractal Image Compression by Multi-Dimensional Nearest Neighbor Search*, IEEE Proceedings - Data Compression Conference – DCC 1995, pp. 222-231, Mar. 1995.

[41] Polvere, M., Nappi, M.. *Speed-up Methods in Fractal Image Coding: Comparison of Methods*, IEEE Transaction on Image Processing, vol. 9, no. 6, pp. 1002-1009, Jun. 2000.

[42] Loganathanff, D., Amudha, J., Mehata, K.M.. *Classification and Feature Vector Techniques to Improve Fractal Image Coding*, IEEE Proceedings - Conference on Convergent Technologies for Asia-Pacific Region – TENCON 2003, vol. 4, pp. 1503 – 1507, Oct. 2003.

[43] El-Khamy, S., Khedr, M., Al-Kabbany, A. *Efficient Fractal Image Coding Using Adaptive Domain Pool Reduction Technique*, IEEE Pacific Rim Conference on Communications, Computers and Signal Processing – PacRim 2007, pp. 62-65, Aug. 2007.

[44] Shannon, E. C., Weaver, W.. *The Mathematical Theory of Communication.* Univ. of Illinois Press, 1949.

[45] Lloyd, S. P.. *Least Squares Quantization in PCM*. IEEE Transactions on. Information Theory, vol. 28, no. 2, pp. 129-137, Mar. 1982.

[46] Huang, C. M., Harris, R. W.. *A Comparison of Several Vector Quantization Codebook Generation Approaches*, IEEE Transactions on Image Processing, vol. 2, no. 5, pp. 108 – 112, Jan. 1993.

[47] Buzo, A., Gray, A. H., Jr, Gray, R. M., Markel, J. D.. *Speech Coding Based upon Vector Quantization*, IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 28, no. 5, pp. 562-574, Oct. 1980.

[48] Sabin, M. J., Gray, R. M.. *Product Code Vector Quantizers for Waveform and Voice Coding*, IEEE Transactions on Signal Processing, vol. 32, no. 3, pp. 474 – 488, Jun. 1984.

[49] Nasrabadi, N. M., Feng, Y.. *Image Processing Using Address-Vector Quantization*, IEEE Transactions on.Communications, vol. 38, no. 12, pp. 2166-2173, Dec. 1990.

[50] Conway, J. H., Sloane, N. J. A.. *Voronoi Regions of Lattices, Second Moments of Polytopes, and Quantization*, IEEE Transactions on Information Theory, vol. 28, no. 2, pp. 211-226, Mar. 1982.

[51] Conway, J. H., Sloane, N. J. A.. *Fast Quantizing and Decoding Algorithms for Lattice Quantizers and Codes*, IEEE Transactions on Information Theory, vol. 28, no. 2, pp. 227-232, Mar. 1982.

[52] Conway, J. H., Sloane, N. J. A.. *On the Voronoi Regions of Certain Lattices*, SIAM Journal of Alg. Disc. Math., 1983.

[53] Haoui, A., Messerschmitt, D. C.. *Predictive Vector Quantization*, Proc. Int. Conf. Acoustics, Speech and Signal Processing, vol. 1, pp. 10.10.1-10.10.4, Mar. 1984.

[54] Moayeri, N., Neuhoff, D. L., Stark, W. E.. *Fine-Coarse Vector Quantization*, IEEE Transactions on Signal Processing, vol. 39, no. 7, pp. 1503 – 1515, Jul. 1991.

[55] Wang, H. S., Moayeri, N.. *Trellis Coded Vector Quantization*, IEEE Transactions on Communications., vol. 40, no. 8, pp. 1273-1276, Aug. 1992.

[56] Khan, M.A.U., Smith, M.J.T., McLaughlin, S.W.. *Jointly optimized trellis-coded residual vector quantization*, IEEE Transactions on Communications, vol. 49, no. 6, pp. 937-942, Jun. 2001.

[57] Fischer, T. R.. *A Pyramid Vector Quantizer*, IEEE Transactions on Information Theory, vol. 32, no. 4, pp. 568-583, Jul. 1986.

[58] Truong, T-K., Jeng, J-H., Reed, I. S., Lee, P. C., Li, A. Q.. *A Fast Algorithm for Fractal Image Compression Using the DCT Inner Product*, IEEE Transactions on Image Processing, vol. 9, no. 4, pp. 529 – 535, Apr. 2000.

[59] Jeng, J. H., Shyu, J. R.. *Fractal Image Compression with Simple Classification Scheme in Frequency Domain*, IEEE Electronics Letters, vol. 36, no. 8, pp. 716 – 717, Apr. 2000.

[60] Bei, C., Gray, R. M.. *An Improvement of The Minimum Distortion Encoding Algorithm for Vector Quantization*, IEEE Transactions on Communications, vol. 33, no. 10, pp. 1132-1133, Oct. 1985.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo