ANSELMO CARDOSO DE PAIVA

UM MÉTODO PARA A COMPRESSÃO DE DADOS VOLUMÉTRICOS BASEADO NA TRANSFORMADA DO COSSENO LOCAL

TESE DE DOUTORADO

DEPARTAMENTO DE INFORMÁTICA

Rio de Janeiro, 25 de abril de 2001.

Anselmo Cardoso de Paiva

Um Método para a Compressão de dados Volumétricos Baseado na Transformada do Cosseno Local

Tese apresentada ao Departamento de Informática da PUC-Rio como parte dos requisitos para a obtenção do título de Doutor em Ciências em Informática - Ênfase: Computação Gráfica.

Orientador: Marcelo Gattass
Co-Orientador: Luiz Velho

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro

Rio de Janeiro, 25 de Abril de 2001.

A meus pais Arnaud e Primavera, minha esposa Fabíola e minha filha Lisle.

"Tudo foi ambicionado. Tudo foi tentado.
O que não consegui fazer eu sonhei fazer."
(Dante Alighieri - A Divina Comédia)

Agradecimentos

Ao Professor Marcelo Gattass pela orientação firme e segura e pelo exemplo como pesquisador.

Ao Professor Jonas Gomes pela cooperação e ajuda na definição deste trabalho e por guiar meus primeiros passos nos caminhos das *wavelets*.

Ao Professor Luiz Velho pela cooperação prestimosa e pelos incentivos na fase final desse trabalho.

Aos demais alunos do grupo de visualização volumétrica do Tecgraf, em especial a André Gerhardt e Marcos Machado, pela cooperação próxima e auxílio na compreensão acerca dos dados sísmicos.

À Fabí e Roseane, pela revisão do texto.

À minha esposa, amiga e companheira Fabí pelo sacrífício da longa ausência de nossa querida terra natal e pelo companheirismo e resignação com que encarou esses anos de trabalho árduo.

A meus irmãos, cunhadas(os), sobrinhos e amigos do Maranhão de quem tive o convívio privado durante o desenvolvimento desse trabalho.

A meu irmão carioca Arlindo Cardaretti, cuja família adotei como minha nessa cidade o que tornou os anos longe da terra natal mais agradáveis.

Aos amigos Andréia, Áurea, Claudinha, Carla, Eduardo Thadeu, Evandro, Turco e demais colegas do Tecgraf.

Aos professores e funcionários do Departamento de Informática.

Ao Tecgraf, pelo excelente ambiente porporcionado para a realização desse trabalho.

Ao Professor Insung Ihm pela disponibilização do conjunto de dados reamostrados do projeto Visible Human.

Aos colegas do Departamento de Informática da UFMA pelo constante incentivo, e pela disponibilidade em se sobrecarregarem com minhas tarefas durante meu afastamento.

À CAPES e à Universidade Federal do Maranhão pelo apoio financeiro.

Resumo

As técnicas de visualização volumétrica apresentam dois clássicos problemas computacionais: necessidade de longo tempo de execução e grande consumo de memória. A popularização das aplicações distribuídas tem despertado um crescente interesse no tratamento dos problemas de grande consumo de memória. Esse problema é fundamental também para garantir o acesso às técnicas de visualização volumétrica em máquinas com pequena capacidade de memória. Nesta tese, propomos um método para a compressão de dados volumétricos baseado na utilização da transformada do cosseno local. Esse método é apropriado ao desenvolvimento de técnicas de visualização volumétrica baseadas na estrutura do dado comprimido. A transformada do cosseno local apresenta a capacidade de possibilitar altas taxas de compressão com a minimização do erro de reconstrução, possibilitando a reconstrução local do volume. Neste trabalho, são apresentados os resultados obtidos com a compressão de volumes reais, estimando a influência dos parâmetros do método de compressão, investigando as possibilidades de adaptação da partição do volume às suas características de freqüências e estimando a capacidade de compressão e os erros introduzidos neste processo.

Abstract

The volume visualization techniques present two classical computer problems: long execution time and large memory requirements. The memory requirements becomes more investigated with the popularization of the distributed applications. This problem is important also, to make possible the access to volume visualization techniques from a personal computer or low end workstation with limited memory. In this thesis we propose a new lossy compression scheme for volumetric data based on the local cosine transform. This method is appropriated for further volume visualization application because it provides good compression rates, minimizes reconstruction errors, and allows local decompression of the volume. We analyze the compression results and estimate some scheme parameter variations, investigating the adaptivity properties and different space decomposition arrangements.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

No escopo deste trabalho, entendemos como dados volumétricos, um conjunto de valores escalares amostrados em uma grade tridimensional, o que na realidade constitui a representação matricial do objeto volumétrico (Gomes *et al.* , 1996). Em geral, esses dados são tratados como uma matriz de elementos de volume, denominados *voxels*. *Voxels* são paralelepípedos, fortemente agrupados, formados pela divisão do espaço do objeto através de um conjunto de planos paralelos aos eixos principais desse espaço. Esses elementos não se interceptam, sendo de tamanho suficientemente pequeno se comparados às características representadas pelos dados volumétricos.

Os dados volumétricos são provenientes de duas fontes principais: simulação e aquisição. Os dados provenientes de simulação são gerados a partir de algum modelo matemático, como em simulações computacionais de mecânica dos fluidos ou na síntese de fenômenos amorfos para a geração de imagens realistas. Por outro lado, uma grande parte dos dados volumétricos é proveniente da conversão de objetos existentes na natureza para uma representação digital (digitalização). Nessa categoria se enquadram os dados gerados pelas várias técnicas de aquisição de imagens médicas e por prospecção sísmica, entre outros. Esses dados são utilizados em um número crescente de aplicações nas áreas científica e de engenharia, as quais tem crescido em número e complexidade. Em geral, trata-se de aplicações envolvendo grandes e complexos conjuntos de dados, os quais precisam ser submetidos a técnicas de visualização volumétrica para extrair informação relevante deles.

As técnicas de visualização de dados volumétricos apresentam os dois clássicos problemas computacionais: necessidade de longo tempo de execução e alto consumo de memória. O problema do tempo de execução foi a principal preocupação dos pesquisadores da área, que desenvolveram várias técnicas de aceleração, e.g. (Levoy, 1990), (Danskin & Hanrahan, 1992), (Yagel & Shi, 1993), e (Lacroute & Levoy, 1995). Por outro lado, o problema do grande consumo de memória recebeu menor atenção durante os primeiros anos do desenvolvimento das técnicas de visualização volumétrica, somente despertando maior interesse de pesquisa com a popularização da Web e das aplicações distribuídas. Mesmo com a queda do preço de memória, e o aumento sucessivo da capacidade de memória dos computadores, esse problema continua relevante; tendo em vista que o tamanho dos conjuntos de dados necessários para as aplicações não pára

de crescer. Conforme (Sayood, 1996), podemos, como um corolário da Primeira Lei de Parkinson [1], afirmar que a necessidade de armazenamento e transmissão de dados se expande pelo menos duas vezes mais rápido que o crescimento dos recursos disponíveis. Assim, verificamos que, sempre que tivermos a limitação de recursos para representar a informação, teremos necessidade de compressão.

O projeto *Visible Human* da Biblioteca Nacional de Medicina dos EUA (*National Library of Medicine* - NLM) é uma importante aplicação na área de visualização volumétrica, impondo grandes desafios quanto à capacidade de memória. Neste projeto, a NLM realizou a aquisição de tomografia computadorizada (CT), ressonância magnética (MRI) e imagens coloridas de seções congeladas (seções criogênicas) de dois cadáveres, um masculino e outro feminino, iniciando a criação de uma biblioteca digital de imagens médicas ((NLM, n.d.) e (Ackerman, 1996)). Nos dados do cadáver masculino, foram adquiridas 1871 imagens transversais em cada modalidade (CT, MRI e seções criogênicas), tomadas a intervalos de 1 mm, resultando em um total de 15 Gbytes de dados volumétricos. As imagens das seções criogênicas do cadáver feminino foram adquiridas com um terço do espaçamento do cadáver masculino, gerando um conjunto de dados de aproximadamente 40 Gbytes. Em um ambiente distribuído de visualização, o simples armazenamento e transmissão desses dados já se apresenta como um grande desafio.

Uma das principais técnicas para geração de imagens da subsuperfície, a área de análise sísmica, apresenta o problema do consumo de memória de forma crítica. Os métodos de análise sísmica são extremamente utilizados na indústria de petróleo para investigar as possibilidades de presença de óleo na subsuperfície. Usualmente, os dados volumétricos manuseados nessa área representam a amplitude sísmica, energia resultante da interação de ondas elásticas, geradas de uma fonte simples, com a geologia da área. Como exemplo do tamanho desses dados, podemos citar o levantamento tridimensional de aproximadamente 800 km^2, relatado em (Brocklehurst, 1995), que necessitou de 2794 Gbytes para armazenamento.

Compressão é um grande desafio para possibilitar a visualização de dados volumétricos em computadores pessoais ou estações de trabalho de baixa capacidade de memória e poder computacional. Nesse caso, se faz necessário um método que permita ao usuário carregar uma versão comprimida do dado volumétrico em uma pequena porção de memória e que possibilite que esse volume comprimido possa ser acessado e visualizado como se o usuário estivesse com o volume descomprimido completamente carregado na memória. Um esquema de compressão como esse deve permitir a des-

[1] Primeira Lei de Parkinson: 'O trabalho se expande para preencher todo o tempo disponível." (Parkinson, 1957)

compressão local dos dados, à medida que eles são necessários para a visualização.

1.1 Trabalhos Anteriores

Na tentativa de diminuir o tamanho dos dados volumétricos, foram utilizadas técnicas de compressão de dados. Inicialmente, foram utilizados os métodos de compressão de imagens, os quais foram aplicados às fatias do volume. Seguindo essa abordagem (Chan *et al.* , 1991) estudou os efeitos da compressão das fatias (imagens) na visualização de dados volumétricos médicos. Esse trabalho foi baseado na transformada discreta do cosseno, e a visualização para aferição dos efeitos da compressão foi realizada com a utilização de métodos de extração de iso-superfícies, através da visualização de imagens obtidas em ângulos oblíquos. Na tentativa de estimar a perda de informação adicionada pelo método de compressão, o trabalho conclui que uma razão de compressão entre 6 e 15 não gera artefatos durante a visualização.

Buscando por uma representação comprimida do dado volumétrico que facilitasse também o processo de visualização, Whilhelms e Van Gelder (Whilhelms & Gelder, 1994) realizaram algumas experiências usando representações hierárquicas, variações da estrutura *octree*. No método proposto, cada nó da *octree* contém um modelo dos dados em termos da representação em uma base de polinômios, uma medida do erro dessa representação e uma medida da importância dessa região na visualização do volume.

O caminhamento na árvore permite a visualização dos dados sem um grande acréscimo no tempo de processamento, através da possibilidade de visualização de uma grande região do volume como um único objeto. Isto é possível quando o erro dessa aproximação é baixo ou a importância da região não é grande. Caso contrário, podemos descer para níveis mais refinados da árvore para gerar um visualização mais apropriada. A compressão é obtida nesse método através do corte da árvore em determinados trechos, o que pode ser feito baseado em um critério de erro de aproximação.

Esse trabalho considerou flexíveis as representações hierárquicas utilizadas, e, embora as mesmas não possuam características ótimas de compressão, podem ser utilizadas para gerar aproximações dos dados volumétricos, sendo particularmente indicadas para dados amostrados em grades irregulares.

Ning e Hesselink (Ning & Hesselink, 1993) propuseram um método que possibilita boas características de compressão e a visualização dos dados volumétricos sem descompressão. Nesse método, o volume é dividido em blocos que são comprimidos usando quantização vetorial, e podem ser visualizados diretamente, sem necessidade de descompressão. No entanto, o processo de visualização usando a representação comprimida do volume resulta em um grande esforço de processamento, e a qualidade das

imagens geradas não é garantida.

Em (Fowler & Yagel, 1994) foi proposto um algoritmo para a compressão sem perdas de dados volumétricos. Esse algoritmo se baseava em uma combinação de DPCM (*Diferential Pulse-Code Modulation*) com codificação de Hufffman (Código Truncado de Huffman). Foi desenvolvido um estimador para dados volumétricos com características de otimalidade em termos de MSE. Os resultados apresentados exibem características de compressão superiores aos algoritmos genéricos de compressão sem perdas, tais como os utilizados no gzip, compress, etc. No entanto, o tempo de processamento se mostrou significativamente superior a esses métodos.

Yeo e Liu (Yeo & Liu, 1995) propuseram um esquema para a visualização de dados volumétricos a partir de uma representação comprimida dos dados. Esse esquema se baseia na blocagem dos dados comprimidos, e na descompressão dos blocos à medida que são necessários. Essa abordagem propicia uma utilização eficiente da memória, permitindo a visualização de volumes que de outro modo não caberiam na memória disponível. O esquema de compressão dos dados é uma adaptação para 3D do algoritmo JPEG, sendo baseado na DCT(Transformada Discreta do Cosseno) e na codificação diferencial seguida por codificação de Huffman para os coeficientes da transformada. Esse método apresenta boas taxas de compressão e tempo razoável para descompressão e visualização do volume, apresentando também os problemas do aparecimento do efeito de bloco quando trabalhamos com altos níveis de compressão.

(Chiueh *et al.* , 1997) desenvolveram um algoritmo para a visualização de dados volumétricos comprimidos, baseado no Teorema da Projeção de Fourier, que representa uma generalização dos métodos propostos por (Dunne *et al.* , 1990) e (Malzbender & Kitson, 1991). No algoritmo proposto, os dados volumétricos são subdivididos em blocos, aos quais é aplicada uma transformada de Fourier, seguida da quantização dos coeficientes e de sua codificação. A quantização é baseada no intervalo dos coeficientes de cada bloco, sendo, em cada um, definido um passo de quantização uniforme. Dentre as vantagens apresentadas por esse método, está a sua capacidade de explorar a coerência local do volume e gerar melhores imagens que os métodos baseados no Teorema da Projeção propostos anteriormente, por aproximar melhor o efeito de oclusão bloco a bloco.

Outra abordagem para visualização de dados volumétricos usando descompressão local foi proposta em (Haley & Blake, 1996). Nesse trabalho foi proposto um algoritmo para a visualização incremental do dado volumétrico, representado em uma estrutura hierárquica (*octree*), como alternativa para a redução dos requisitos de memória do algoritmo de visualização volumétrica *Shear Warping* (Lacroute & Levoy, 1995).

Procurando um domínio de representação que fornecesse boas características de compressão e tornasse possível o desenvolvimento de algoritmos de visualização eficientes, foram desenvolvidas algumas técnicas baseadas no domínio espaço×freqüência. Essas técnicas procuram explorar as características do novo domínio, tais como boa localização de freqüência ou suporte compacto das bases. Uma característica comun a esses métodos é a utilização de reconstrução local do volume, sem necessitar de sua descompressão total.

Muraki ((Muraki, 1992) e (Muraki, 1993)) introduziu a utilização de *wavelets* para aproximar de maneira eficiente os dados volumétricos. Tratando os coeficientes de *wavelets* de pequeno valor como zero, pode ser verificado que o volume é descrito por um número relativamente pequeno de coeficientes. No entanto, nesses trabalhos não foi feita nenhuma referência sobre o tempo necessário para a codificação, a possibilidade de acesso randômico aos coeficientes ou a quantidade de memória necessária para a visualização.

Em (Westermann, 1994) é apresentada uma abordagem para a visualização de volumes baseada na transformada de *wavelets*, demonstrando como resolver a integral de *rendering* diretamente no domínio das frequências. Assim, a visualização pode ser realizada sem necessidade de calcular a transformada inversa do volume. Também são apresentadas técnicas para aceleração do algoritmo de visualização baseadas nas propriedades de multiresolução da representação por *wavelets*. Como a representação no domínio de *wavelets* é esparsa, podemos utilizar um pequeno número de coeficientes para representar o volume, comprimindo-o. Verificou-se que o tempo de visualização depende diretamente do suporte da *wavelet* utilizada, o que também influencia significativamente a capacidade de compressão da transformada. Logo, para altas taxas de compressão, o tempo para a visualização não é satisfatório.

(Gross *et al.* , 1998) apresenta métodos de compressão de dados volumétricos baseados na representação em bases de *wavelets* do volume já classificado. Esses métodos apresentam altas taxas de compressão, com a grande desvantagem de não permitir a reclassificação do volume, tarefa essencial na interação com dados volumétricos.

(Grosso *et al.* , 1996) propôs uma estrutura de dados para representações esparsas, que não explora todo o potencial de compressão, mas que pode ser combinada com um método inteligente de caminhamento sobre a pirâmide de multiresolução dos coeficientes de *wavelets* para acelerar a visualização de volumes representados nessas bases.

Em (Thoma & Long, 1997) foram descritos resultados experimentais comparando a aplicação aos dados do projeto *Visible Human* de várias técnicas de compressão com perda. Os resultados desse trabalho apontaram os métodos baseados em *wavelets*

como os melhores.

Seguindo essa tendência, Ihm e Park (Ihm & Park, 1998) propuseram um esquema para a compressão de grandes volumes de dados, baseados na transformada de *wavelet*. O esquema proposto se baseia na partição do volume em blocos de 16^3 *voxels*, os quais são transformados para uma representação de *wavelets*. Em seguida, os coeficientes são submetidos a um processo de limiarização com um valor τ, sendo então codificados. Na codificação, os coeficientes são organizados de modo a permitir o acesso randômico a qualquer coeficiente do bloco. Isto favorece o desenvolvimento de algoritmos de visualização adequados à estrutura do dado comprimido, os quais necessitam apenas da descompressão local do volume. O esquema proposto resulta em boas características de compressão.

Seguindo essa linha de trabalho (Kim & Shin, 1999) desenvolveram um método de compressão que permite a visualização de grandes volumes de dados, os quais são divididos em blocos de 8^3 *voxels*. Em seguida, os coeficientes são normalizados e reordenados de acordo com a ordem de reconstrução. A estrutura de dados proposta em (Grosso *et al.* , 1996) é utilizada, ou seja, os coeficientes são codificados por RLE (*Run Length Encoding*). Também foi desenvolvida uma estrutura (*cache*) com o objetivo de acelerar o processo de visualização a partir dos dados comprimidos. Esse trabalho apresenta boas características de compressão dos dados, permitindo a visualização em tempos aceitáveis do dado comprimido.

(Rodler, 1999) propôs um método, baseado na transformada de *wavelets* e no algoritmo de codificação por subbandas tridimensional desenvolvido para codificação de seqüências de vídeos por (Chen & Pearlman, 1996). O algoritmo proposto possibilita o acesso randômico aos *voxels* do volume, e altas taxas de compressão. Por outro lado, possui o problema de alto custo computacional para a descompressão. Os métodos baseados em *wavelets* apresentam dificuldade para representar características períodicas nos sinais.

Na busca por métodos de compressão sem perdas, adequados à utilização em volumes médicos para diagnóstico direto, uma série de trabalhos foram realizados, sem nenhuma preocupação com um possível processo de visualização tridimensional do volume. (Knittel, 1995) propôs a utilização de BTC (*Block truncation Code*) para a compressão de dados volumétricos médicos, que permitia a compressão de blocos de 12 *voxels* em 32 bits de maneira redundante. (Menegaz *et al.* , 1999) propõem um método de compressão sem perda baseado em *wavelets* que explora a completa distribuição dos dados, e permite a reconstrução de uma fatia do volume sem necessidade da completa reconstrução do volume. O algoritmo proposto é baseado na codificação LZC (Taubman & Zakhor, 1994).

Em (Kim & Pearlman, 1999) é proposto um método de compressão sem perda, baseado na transformada de *wavelets* tridimensional. Este método é uma aplicação do algoritmo SPIHT (*Set Partitioning in Hierarchical Trees*) (Said & Pearlman, 1996) a dados volumétricos médicos. A transformada de *wavelets* é implementada por *lifting*, com a utilização de filtros inteiros para manter a transformada unitária. Os resultados mostraram melhor performance que os métodos anteriores.

Na área de representação de sinais (funções), vários trabalhos independentes ((Malvar, 1988), (Coifman *et al.* , 1991), (Princen & Bradley, 1986), (Daubechies & Lagarias, 1991) e (Laeng, 1990)) descobriram uma maneira de construir uma base ortogonal diferenciável, subordinada a partições arbitrárias da reta real. Malvar mostrou, em (Malvar, 1988) e (Malvar, 1990), que era possível criar bases ortogonais discretas com janelas diferenciáveis moduladas por bases cosseno IV. Esse resultado foi redescoberto para funções contínuas por Coifman e Meyer (Coifman *et al.* , 1991). Essa construção é conhecida por várias denominações diferentes: transformada do cosseno sobreposto (*lapped cosine transform* - LCT), transformada ortogonal sobreposta (*lapped orthogonal transform* - LOT), transformada ortogonal sobreposta modulada (*modulated lapped orthogonal transform*), bancos de filtros com cancelamento de *alias* no domínio do tempo (*time-domain aliasing cancellation filter banks*), *wavelets* de Malvar, transformada do cosseno local (*local cosine transform*), entre outros. Nesta tese usaremos a denominação transformada do coseno local ou o acrônimo LCT.

Em (Malvar & Staelin, 1989) e (Malvar, 1990) a transformada do cosseno local foi utilizada para compressão de sinais unidimensionais de fala humana, resultando na redução do denominado efeito de blocagem e com ganhos na codificação por transformada (*Transform Coding Gain*-TCG) similares aos da transformada discreta do cosseno (DCT).

(Wickerhauser, 1992) discute duas abordagens alternativas ao algoritmo JPEG, padrão para compressão de imagens. Uma das alternativas é um algoritmo de compressão baseado na transformada do cosseno local, usado com a abordagem de busca da melhor representação para o sinal (*best basis*). Comparando as curvas razão de compressão $\times$ distorção das imagens comprimidas, foi concluído que a altas taxas de compressão os resultados fornecidos pelo algoritmo JPEG com a transformada do cosseno local se comporta melhor que os obtidos com a utilização conjunta da transformada de *wavelets* com o algoritmo *best basis*. Para baixas taxas de compressão, os resultados se inverteram. Essa comparação também é feita em (Wickerhauser *et al.* , 1994), para fluxos turbulentos de fluidos em duas dimensões.

Duas variações da transformada do cosseno local (usando DCT-II e DCT-IV) foram utilizadas em (Aharony *et al.* , 1993) e (Aharony *et al.* , 1994) dentro do

algoritmo JPEG para reduzir os efeitos de blocagem. Os resultados demonstraram que a abordagem com a transformada do cosseno local resulta em menor erro médio quadrático (MSE) que o algoritmo JPEG original, e os resultados utilizando a DCT-II apresentavam melhor qualidade nas imagens reconstruídas, embora apresentassem menor capacidade de compressão.

Jawerth e Sweldens (Jawerth *et al.* , 1994) generalizaram a construção original de Malvar, Coifman e Meyer para as bases de cossenos locais, para adicionar a vantagem de boa representação para componentes lineares ou constantes, o que reduz o efeito de blocagem e melhora a razão sinal ruído do sinal comprimido. Essas construções modificadas foram aplicadas à compressão de imagens demonstrando suas possibilidades.

Mais recentemente, a transformada do cosseno local foi aplicada a dados sísmicos unidimensionais ((Wang *et al.* , 1998), (Meyer, 1999) and (Wu & Wang, 1999)). Em (Wang *et al.* , 1998) foi concluído que mesmo no caso de altas taxas de compressão (e.g. 40:1) as características mais importantes dos dados se mantinham preservadas, sendo possível a reconstrução de imagens sísmicas de alta qualidade.

1.2 Contribuição

Esta tese apresenta um novo algoritmo para a compressão de dados volumétricos, baseado na transformada do cosseno local. Esta proposta se encaixa numa tendência da área de pesquisa de visualização volumétrica que busca a representação compacta de dados volumétricos em uma estrutura que seja apropriada ao desenvolvimento posterior de algoritmos de visualização que possam tratar esses dados sem descomprimi-los, ou realizando apenas uma descompressão do dado a medida que cada porção do mesmo é necessária.

A transformada do cosseno local é uma transformada de classe C^{∞}, com base ortonormal, suporte compacto e que fornece uma representação em escala do dado. Além disso, por ser uma transformada ortogonal que possui sobreposição das funções da base, é adequada ao desenvolvimento de um esquema de compressão por blocos, minimizando o efeito de bordas, e conseqüentemente os erros de reconstrução. Por ser baseada na transformada do cosseno, a transformada do cosseno local apresenta boas características de compressão.

Um algoritmo para a reconstrução local do volume, bloco a bloco, foi desenvolvido, o que propicia fácil implementação de algoritmos de visualização volumétrica para a representação comprimida dos dados, principalmente algoritmos baseados no espaço dos objetos.

Também é apresentada nesta tese uma investigação de vários esquemas para

subdivisão do dado volumétrico, e de seus reflexos no esquema de compressão.

1.3 Organização da Tese

Esta tese está organizada da seguinte maneira. No capítulo seguinte, apresentamos a teoria matemática que leva à definição da transformada do cosseno local, e suas propriedades. Neste capítulo, mostramos como é desenvolvido o algoritmo rápido para cálculo da transformada e também apresentamos o algoritmo de *best basis* que permite adaptar a partição do volume às suas características de freqüência, encontrando a melhor representação do volume nas bases de cossenos locais.

No capítulo 3, apresentamos uma introdução sobre compressão de dados. Apresentamos os principais métodos de compressão, detalhando os métodos baseados em transformadas, os mais usados para a compressão de imagens e volumes.

No capítulo 4 apresentamos o esquema de compressão proposto nesta tese, justificando algumas das decisões tomadas ao longo de seu desenvolvimento. Neste capítulo, apresentamos os resultados iniciais obtidos com os volumes testados, mostrando o efeito de variação de alguns parâmetros da transformada no resultado final. São mostrados também os resultados que caracterizam o comportamento da transformada mostrando a influência da variação de alguns parâmetros. Finalmente, são apresentados os resultados obtidos pelo algoritmo de compressão desenvolvido quando aplicado a dados reais, para em seguida, no capítulo 5, apresentarmos as conclusões finais do trabalho e as futuras direções de pesquisa.

Capítulo 2

A Transformada do Cosseno Local

No estudo de uma representação para um sinal que apresente a menor autocorrelação possível, por exemplo, para usar essa representação em um método de compressão de sinais, podemos verificar que a correlação presente nos sinais é geralmente um fenômeno local. Assim, observamos que as bases de Fourier não são adequadas para esse tipo de representação, em razão das funções dessa base serem não locais, ou seja, o suporte das funções da base de Fourier se estende por todo o domínio. Para obter melhor representação dos fenômenos locais dos sinais, melhor localização das freqüências, podemos dividir o eixo do tempo em intervalos e calcular a base de Fourier para cada intervalo, separadamente. Esta abordagem possui as desvantagens da perda de correlação entre os intervalos, onde a base de Fourier é calculada, e de propiciar o aparecimento dos chamados efeitos de borda ou efeitos de blocagem. Esses efeitos de borda geram, na representação por bases de Fourier, coeficientes de grande magnitude em razão dos sinais analisados serem não periódicos. A não periodicidade dos sinais é interpretada pela análise de Fourier como uma descontinuidade ou uma variação brusca no sinal. A introdução das bases trigonométricas locais ((Malvar, 1988), (Coifman *et al.* , 1991)) é um melhoramento dessa idéia. Essas bases utilizam funções trigonométricas (e.g. cosseno) como funções da base em substituição à base de Fourier, as quais são multiplicadas por uma função janela diferenciável. Dessa maneira, conseguimos maior correlação local na representação do sinal, minimizando o aparecimento dos efeitos de borda e, adicionalmente, com liberdade para a escolha da partição do eixo do tempo que melhor represente as características do sinal. Nas seções seguintes serão apresentadas as maneiras utilizadas para gerar bases de cossenos locais e o método para encontrar o conjunto de bases que melhor representa um dado sinal.

2.1 Bases de Cossenos

Seja $f \in L^2[0,1]$ e $f(0) \neq f(1)$ uma função diferenciável, então a base de cosseno I pode atenuar os efeitos de borda através da restauração de uma extensão periódica $\widehat{f}$ de f (veja Figura 2.1), que é contínua se a função f for contínua. Dessa forma, as altas freqüências introduzidas pela interpretação de uma mudança brusca nas bordas são menores que na base de Fourier.

A função $\widehat{f}$, que é simétrica em torno de 0, igual a f sobre o intervalo $[0,1]$ e

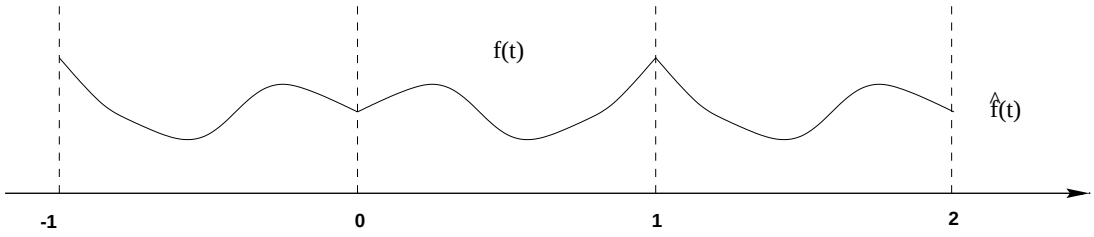


Figura 2.1: Extensão cosseno I para uma função $f(t)$.

possui período 2, é dada por :

$$\widehat{f}(t) = \begin{cases} f(t) & \text{para } t \in [0,1] \\ f(-t) & \text{para } t \in (-1,0] \end{cases} \tag{2.1}$$

Escrevendo a expansão na base de Fourier de $\widehat{f}$ sobre o intervalo $[0,2]$ como uma soma de senos e cossenos, podemos verificar que todos os termos seno são zero (a função $\widehat{f}$ é par). Observando que $\widehat{f}(t) = f(t)$ sobre o intervalo $[0,1]$, temos que f pode ser escrita como uma soma de cossenos. Assim, a família de funções

$$\left\{\lambda_k \sqrt{2}\cos(\pi k t)\right\}_{k\in\mathbb{N}} \qquad \text{with } \lambda_k = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } k = 0 \\ 1 & \text{if } k \neq 0 \end{cases} \tag{2.2}$$

é uma base ortonormal de $L^2[0,1]$, denominada base cosseno I.

Outras bases de cosseno são geradas a partir de diferentes extensões de f sobre o intervalo $[0,1]$. Estaremos especialmente interessados nas bases cosseno IV, as quais são utilizadas para construir bases de cossenos locais com janelas suaves. As bases cosseno IV aparecem quando a função f é estendida em uma função $\widehat{f}$ de período 4 (mostrada na Figura 2.2), simétrica em torno de 0 e antisimétrica em torno de 1 e -1, sendo descritas por:

$$\widehat{f}(t) = \begin{cases} f(t) & \text{se } t \in [0,1) \\ f(-t) & \text{se } t \in (-1,0) \\ -f(2-t) & \text{se } t \in [1,2) \\ -f(2+t) & \text{se } t \in [-1,-2) \end{cases} \tag{2.3}$$

Escrevendo a expansão em série de Fourier de $\widehat{f}$, e realizando análise similar à realizada para a base cosseno I, podemos concluir que a base cosseno IV é representada

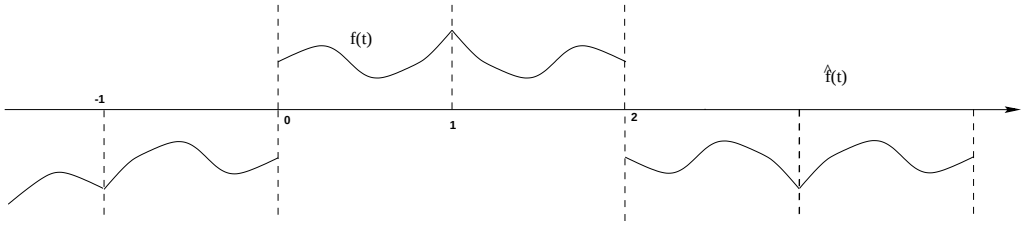


Figura 2.2: Extensão cosseno IV de uma função $f(t)$.

pela família $\left\{\sqrt{2}\cos\left(\left(k+\frac{1}{2}\right)\pi t\right)\right\}_{k\in\mathbb{N}}$, sendo uma base ortonormal de $L^2[0,1]$.

Essas bases possuem problemas similares aos apresentados pelas bases de Fourier, i.e. os coeficientes de uma função suave possuem decaimento lento em altas freqüências. Isto acontece em razão da decomposição corresponder a uma extensão descontínua de f em cada intervalo. A importância dessas bases é dada por seu uso na construção de bases de cossenos locais que não apresentam esse problema.

As bases discretas do cosseno são derivadas das bases discretas de Fourier com a mesma abordagem usada no caso contínuo. Assim, temos a família

$$\left\{\sqrt{2}\cos\left[\frac{\pi}{N}\left(k+\frac{1}{2}\right)\left(n+\frac{1}{2}\right)\right]\right\}_{0\leq k<N}, \tag{2.4}$$

que representa a base ortonormal de $\mathbb{C}^N$, conhecida como base discreta do cosseno IV.

A transformada associada com essa base é denominada DCT-IV (*Discrete Cosine IV Transform*), e a transformada discreta $F(u)$ de uma função $f(j)$, $j = 0,1,\ldots,N-1$ é definida como:

$$F(u) = \sqrt{\frac{2}{N}}\sum_{n=0}^{N-1} f(j)\cos\left[\frac{\pi}{N}\left(j+\frac{1}{2}\right)\left(u+\frac{1}{2}\right)\right] \qquad u = 0,1,\ldots,N-1. \tag{2.5}$$

A inversa da DCT-IV é definida como

$$f(j) = \sqrt{\frac{2}{N}}\sum_{n=0}^{N-1} F(u)\cos\left[\frac{\pi}{N}\left(j+\frac{1}{2}\right)\left(u+\frac{1}{2}\right)\right] \qquad j = 0,1,\ldots,N-1. \tag{2.6}$$

A extensão para dimensões maiores que 1 pode ser feita observando que a transformada é separável, portanto podemos aplicar um produto tensorial. Seguindo essa abordagem, podemos afirmar que, para três dimensões, a transformada discreta

do cosseno IV é definida por:

$$F(u,v,w) = \frac{2}{N}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\sum_{k=0}^{N-1} f(i,j,k)\cos\left[\frac{\pi}{N}\left(i+\frac{1}{2}\right)\left(u+\frac{1}{2}\right)\right] \cos\left[\frac{\pi}{N}\left(j+\frac{1}{2}\right)\left(v+\frac{1}{2}\right)\right] \cos\left[\frac{\pi}{N}\left(k+\frac{1}{2}\right)\left(w+\frac{1}{2}\right)\right], \tag{2.7}$$

onde $u, v, w = 0, 1, \ldots, N-1$.

Em razão da propriedade de separabilidade, temos que qualquer transformada k-dimensional pode ser calculada através do cálculo de k aplicações sucessivas da transformada unidimensional ou sua inversa. Logo, podemos calcular a transformada de um volume (transformada 3D) aplicando a DCT-IV unidimensional nas linhas do volume, nas colunas e ao longo de sua profundidade.

Uma das desvantagens desta transformada para a compressão é o fato da mesma não possuir um coeficiente representando o valor médio do sinal transformado. Esta propriedade é denominada resolução de constante.

Podemos relacionar o cálculo da DCT-IV de uma função discreta real com N amostras, com o cálculo da transformada de Fourier de uma função discreta complexa com $\frac{N}{2}$ amostras. Isto é feito através da utilização de uma fórmula introduzida em (Duhamel *et al.* , 1991). A DCT-IV pode ser calculada dividindo a função discreta real $f[n],\ n = 0, 1, \ldots, N-1$ em duas funções b e c, cada uma com a metade das amostras. Essa divisão é feita de modo que uma delas esteja associada aos valores das amostras de índices ímpares e a outra associada às amostras de índices pares, como mostrado abaixo:

$$b[n] = f[2n] \tag{2.8}$$

$$c[n] = f[N-1-2n]. \tag{2.9}$$

Para $0 \leq n < \frac{N}{2}$ podemos gerar uma outra função complexa $g[n]$ dada por:

$$g[n] = (b[n] + ic[n])\exp\left[-i\left(n+\frac{1}{4}\right)\frac{\pi}{N}\right]. \tag{2.10}$$

Finalmente, podemos afirmar que, se $\widehat{g}[k]$ é a transformada discreta de Fourier

de $g[n]$, então a DCT-IV de $f[n]$ é $\widehat{f}_{IV}[k]$ dada por

$$\widehat{f}_{IV}[2k] = Real\left\{\exp\left[\frac{-i\pi k}{N}\right]\widehat{g}[k]\right\}, \tag{2.11}$$

$$\widehat{f}_{IV}[N-2k-1] = -Im\left\{\exp\left[\frac{-i\pi k}{N}\right]\widehat{g}[k]\right\}. \tag{2.12}$$

Assim, podemos ver que uma implementação do algoritmo de cálculo da DCT-IV, baseada em uma implementação eficiente da FFT (transformada rápida de Fourier), requer somente $O(nlogn)$ operações. Este algoritmo é denominado transformada rápida DCT-IV.

2.2 Base de Cossenos em Blocos

Bases de cossenos locais são bases em blocos que segmentam o eixo do tempo em intervalos, cujos tamanhos são adaptados às estruturas dos sinais. Esses blocos são obtidos através da multiplicação de janelas escaladas e transladadas ao longo desse eixo com as funções cosseno.

Bases ortogonais em blocos são obtidas dividindo o eixo do tempo em intervalos consecutivos $[a_p, a_{p+1}]$, com tamanhos arbitrários $l_p = a_{p+1} - a_p$. Dada a função $g = 1_{[0,1]}$, temos que um intervalo arbitrário é recoberto pela janela retangular escalada

$$g_p(t) = 1_{[a_p, a_{p+1}]}(t) = g\left(\frac{t-a_p}{l_p}\right). \tag{2.13}$$

Assim, se temos uma base ortonormal $\{e_k\}_{k\in\mathbb{Z}}$ de $L^2[0,1]$, então

$$\left\{g_{p,k}(t) = g_p(t)\frac{1}{\sqrt{l_p}}e_k\left(\frac{t-a_p}{l_p}\right)\right\}_{(p,k)\in\mathbb{Z}} \tag{2.14}$$

é uma base ortonormal em blocos de $L^2(\mathbb{R})$.

No caso discreto, temos o intervalo definido como $[a_p, a_{p+1}-1]$, assumindo que $\{e_{k,l}\}_{0\le k\le l}$ é uma base ortonormal de $\mathbb{C}^l$, para qualquer $l > 0$, então a família

$$\left\{g_{p,k}[n] = g_p[n]e_{k,l_p}[n-a_p]\right\}_{\substack{0\le k\le l_p \\ p\in\mathbb{Z}}} \tag{2.15}$$

é uma base ortonormal em blocos de $l^2(\mathbb{Z})$.

Quando usamos funções cosseno para construção de bases em bloco, geramos

uma família $\{g_{pk}(t)\}_{\substack{k\in\mathbb{N}\\ p\in\mathbb{Z}}}$, onde

$$g_{pk}(t) = g_p(t)\sqrt{\frac{2}{l_p}}cos\left(\pi k\frac{t-a_p}{l_p}\right) \tag{2.16}$$

é uma base de cosseno em blocos para $L^2(\mathbb{R})$.

Bases de cossenos em blocos para sinais 2D (imagens) são construídas através da partição do espaço $\mathbb{R}^2$ em intervalos retangulares $\{[a_p, b_p]\times[c_q, d_q]\}_{p,q\in\mathbb{Z}}$, os quais possuem tamanho e largura arbitrários dados por: $lp = b_p - a_p$ e $w_q = d_q - c_q$. Assim, a família $\{g_{p,q,k,j}\}_{(k,j)\in\mathbb{Z}^2}$, onde

$$g_{p,q,k,j}(x,y) \;=\; g_p\left(\frac{x-a_p}{l_p}\right) g_q\left(\frac{y-c_q}{w_q}\right)\frac{1}{\sqrt{l_p w_q}}\cos\left(\pi k\frac{x-a_p}{l_p}\right)\cos\left(\pi j\frac{y-c_q}{w_q}\right) \tag{2.17}$$

é uma base 2D de cossenos em bloco.

Similarmente, temos que as bases de cossenos em bloco para sinais 3D (volumes) são geradas pela família, $\{g_{p,q,r,k,j,i}\}_{(k,j,i)\in\mathbb{Z}^3}$ dada por

$$\begin{aligned} g_{p,q,r,k,i,j}(x,y,z) = {} & g_p\left(\frac{(x-a_p)}{l_p}\right) g_q\left(\frac{(y-c_q)}{w_q}\right) g_r\left(\frac{(z-f_r)}{s_r}\right) \\ & \frac{1}{l_p w_q s_r}\cos\left(\pi k\frac{x-a_p}{l_p}\right)\cos\left(\pi i\frac{y-c_q}{w_q}\right)\cos\left(\pi j\frac{z-f_r}{s_r}\right), \end{aligned} \tag{2.18}$$

na qual os intervalos volumétricos são definidos por $\{[a_p, b_p]\times[c_q, d_q]\times[f_r, h_r]\}$, onde $l_p = b_p - a_p$, $w_q = d_q - c_q$ e $s_r = h_r - f_r$ definem o tamanho dos intervalos ao longo dos três eixos principais do volume.

Essas bases em bloco são construídas com janelas retangulares descontínuas, as quais particionam o espaço em intervalos disjuntos, gerando artefatos devidos a essas descontinuidades, e.g. diferenças na reconstrução em torno de um ponto oriundas de dois intervalos adjacentes naquele ponto. Para tentar evitar essas descontinuidades é necessária a utilização de janelas diferenciáveis com decaimento suave nas bordas.

2.3 Bases de Cossenos Locais

Vários trabalhos independentes ((Malvar, 1988), (Coifman *et al.* , 1991), (Princen & Bradley, 1986), (Daubechies & Lagarias, 1991), (Laeng, 1990)) descobriram a maneira de construir bases ortogonais diferenciáveis subordinadas a partições arbitrárias da reta real.

Malvar mostrou em (Malvar, 1988) e (Malvar, 1990) que era possível a criação de bases ortogonais discretas com janelas contínuas diferenciáveis moduladas por uma base cosseno-IV. Coifman e Meyer (Coifman *et al.* , 1991) redescobriram esse resultado para funções contínuas.

Seja I_p uma partição do eixo do tempo em intervalos que se sobrepõem uns aos outros:

$$I_p = [a_p - \eta_p, a_{p+1} + \eta_{p+1}], \tag{2.19}$$

com $\lim_{p\to-\infty} a_p = -\infty$ e $\lim_{p\to+\infty} a_p = +\infty$, de modo que I_{p-1} e I_{p+1} não se interceptam. Podemos considerar que cada intervalo I_p é dividido em subintervalos. Dois desses subintervalos, O_p e O_{p+1}, representam a porção em que há sobreposição na borda direita e esquerda (como mostrado na Figura 2.3) e o terceiro representa um intervalo central C_p. Esses subintervalos podem ser definidos por:

$$O_p = [a_p - \eta_p, a_p + \eta_p], \tag{2.20}$$

$$C_p = [a_p + \eta_p, a_{p+1} - \eta_{p+1}]. \tag{2.21}$$

Podemos definir uma função de janelamento (também denominada função sino) g_p, cujo suporte é I_p, e que possui perfil decrescente em O_p e perfil crescente em O_{p+1}, dada por

$$g_p(t) = \begin{cases} 0 & \text{se } t \notin I_p, \\ \beta\left(\frac{t-a_p}{\eta_p}\right) & \text{se } t \in O_p, \\ 1 & \text{se } t \in C_p, \\ \beta\left(\frac{a_{p+1}-t}{\eta_{p+1}}\right) & \text{se } t \in O_{p+1}, \end{cases} \tag{2.22}$$

onde $\beta(t)$ é uma função perfil monotônica, de modo que:

$$\beta(t) = \begin{cases} 0, & \text{se } t < -1, \\ 1, & \text{se } t > -1, \end{cases} \tag{2.23}$$

$$\forall t \in [-1, 1] \quad \beta^2(t) + \beta^2(-t) = 1. \tag{2.24}$$

Agora, podemos definir uma base de cossenos locais de $L^2(\mathbb{R})$ como a base derivada da base de cosseno-IV de $L^2[0, 1]$, através da multiplicação de versões transladadas e escaladas de cada função da base de cossenos com a função de janelamento

g_p. Assim, a família de funções locais de cosseno

$$\left\{ g_{p,k}(t) = g_p(t)\sqrt{\frac{2}{l_p}}\cos\left(\pi\left(k+\frac{1}{2}\right)\frac{t-a_p}{l_p}\right)\right\}_{\substack{k\in\mathbb{N}\\ p\in\mathbb{Z}}} \tag{2.25}$$

é uma base ortonormal de $L^2(\mathbb{R})$.

Essas bases são compostas de funções $g_{p,k}(t)$, com suporte compacto em

$$[a_p - \eta_p, a_{p+1} + \eta_{p+1}]\,, \tag{2.26}$$

como mostrado na Figura 2.3.

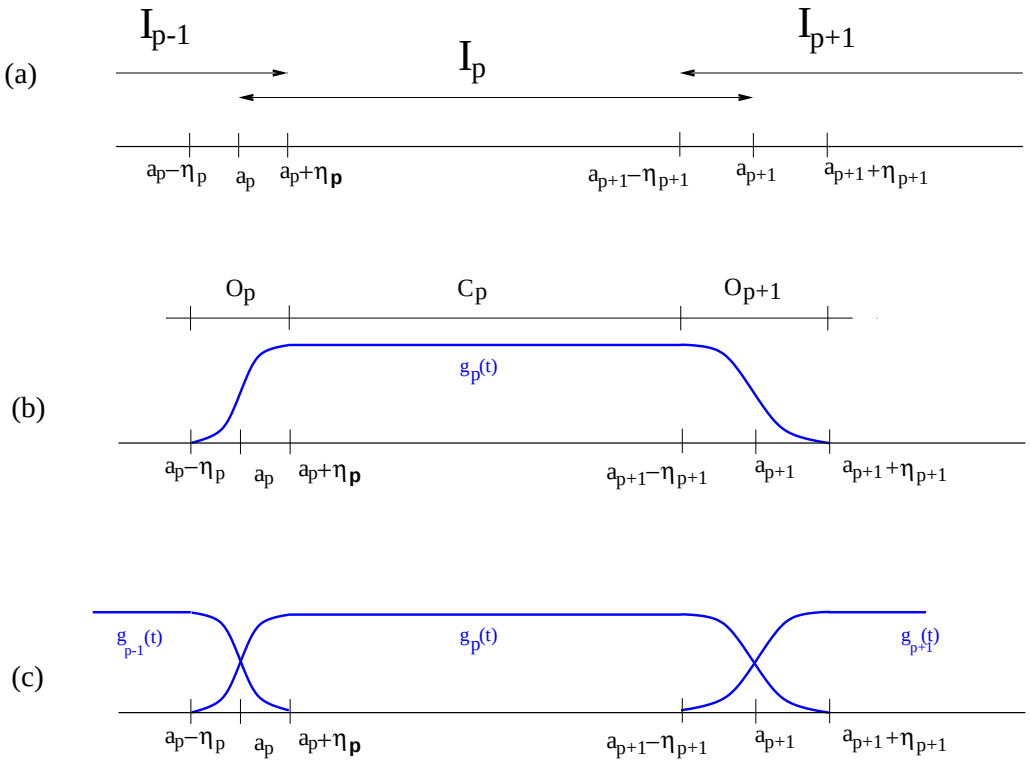


Figura 2.3: (a) Intervalos sobrepostos recobrindo o eixo do tempo. (b) A função de janelamento $g(t)$ e os subintervalos. (c) As janelas sobrepostas dos intervalos $p-1$, p e $p+1$.

A regularidade da função de janelamento g_p depende da regularidade da função de perfil β utilizada para definí-la. Uma construção geral para funções de perfil é dada em (Auscher *et al.* , 1992). No entanto, podemos observar que qualquer função

satisfazendo as equações 2.23 deve ser da forma

$$\beta(t) \stackrel{def}{=} \exp\left[i\rho(t)\right] \sin\left[\theta(t)\right], \tag{2.27}$$

onde ρ e θ são funções reais satisfazendo

$$\rho(t) = \begin{cases} 2n\pi, & \text{se } t < -1, \\ 2m\pi, & \text{se } t > 1. \end{cases} \tag{2.28}$$

$$\theta(t) = \begin{cases} 0, & \text{se } t < -1, \\ \frac{\pi}{2}, & \text{se } t > 1. \end{cases} \tag{2.29}$$

$$\theta(t) + \theta(-t) = \frac{\pi}{2}. \tag{2.30}$$

Como exemplo de uma função com estas propriedades, obtida fazendo $\rho = 0$, temos a equação abaixo, que foi utilizada durante o desenvolvimento deste trabalho.

$$\beta(t) = \begin{cases} 0 & \text{para } t \leq -1, \\ \sin\left(\frac{\pi}{4}(1+t)\right) & \text{para } t \in [-1,1], \\ 1 & \text{para } t \geq 1. \end{cases} \tag{2.31}$$

Podemos obter funções reais continuamente diferenciáveis d vezes ($d = 2^k - 1$), para números d arbitrariamente grandes, através da substituição repetida de t por $\sin\frac{\pi t}{2}$ na equação abaixo

$$\beta_{k+1}(t) = \beta_k\left(\sin\frac{\pi t}{2}\right), \text{ for } t \in [-1,1]. \tag{2.32}$$

Uma base de cossenos locais pode ser representada graficamente como um conjunto de retângulos recobrindo todo o plano tempo×freqüência, como mostrado na Figura 2.4. Isto gera uma grade nesse plano, cujas células possuem tamanhos variando no tempo. Cada retângulo é definido por

$$[a_p, a_{p+1}] \times \left[\xi_{p,k} - \frac{\pi}{2l_p}, \xi_{p,k} + \frac{\pi}{2l_p}\right], \tag{2.33}$$

com $\xi_{p,k} = \frac{\pi\left(k+\frac{1}{2}\right)}{l_p}$. Analisando a transformada de Fourier da função de janelamento, podemos verificar que as funções locais de cosseno possuem freqüências bem localizadas. Isto advém do fato da transformada de Fourier de g_p ser formada por dois picos modulados centralizados em $\xi_{p,k}$ e $-\xi_{p,k}$.

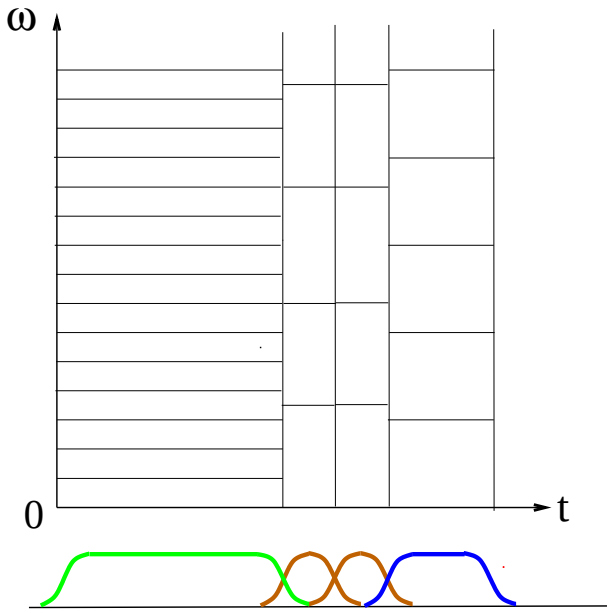


Figura 2.4: Decomposição do domínio tempo–freqüência em vetores de cossenos locais.

Bases de cossenos locais são discretizadas através da substituição da base ortogonal de $L^2[0,1]$ por uma base discreta de cossenos-IV e da amostragem uniforme da janela g_p. Logo, se tivermos uma janela discreta $g_p[n]$, então a família

$$\left\{ g_{p,k}[n] = g_p[n]\sqrt{\frac{2}{l_p}}\cos\left[\pi\left(k+\frac{1}{2}\right)\frac{n-a_p}{l_p}\right]\right\}_{\substack{0\le k<l_p \\ p\in\mathbb{Z}}} \tag{2.34}$$

é uma base ortonormal de $l^2(\mathbb{Z})$.

Para decompor uma função f na base de cossenos locais, devem ser calculados os produtos internos da função com os vetores discretos da base ($\langle f, g_{p,k}\rangle$). Um algoritmo rápido, introduzido em (Malvar, 1992), substitui os cálculos de $\langle f, g_{p,k}\rangle$ por cálculos de produtos internos na base original, que podem ser calculados com o algoritmo rápido da DCT-IV, adicionado a um procedimento de dobragem (*folding*). Suponha que desejamos dobrar a função f em torno do ponto a_p, ao longo dos intervalo $[a_p-\eta_p, a_p)$ e $(a_p, a_p+\eta_p]$, usando a função de janelamento g_p. Temos que a função dobrada (*folded*) é dada por:

$$f_p^{fold}(x) = \begin{cases} g_p(x)f(x) + g_p(2a_p-x)f(2a_p-x) & \text{se } x\in O_p, \\ f(x) & \text{se } n\in C_p, \\ g_p(x)f(x) - g_p(2a_{p+1}-x)f(2a_{p+1}-x) & \text{se } x\in O_p. \end{cases} \tag{2.35}$$

A operação de dobragem pode ser representada graficamente através do espelhamento da porção da função modulada (Figura 2.5(b)), que se encontra sobre o intervalo $[a_p - \eta_p, a_p]$ ($[a_{p+1}, a_{p+1} + \eta_{p+1}]$), em torno do eixo $t = a_p$ ($t = a_{p+1}$)(Figura 2.5c), seguido da adição da porção espelhada à função com suporte no subintervalo $[a_p, a_p + \eta_p]$ ($[a_{p+1}, a_{p+1} + \eta_{p+1}]$) (Figura 2.5d).

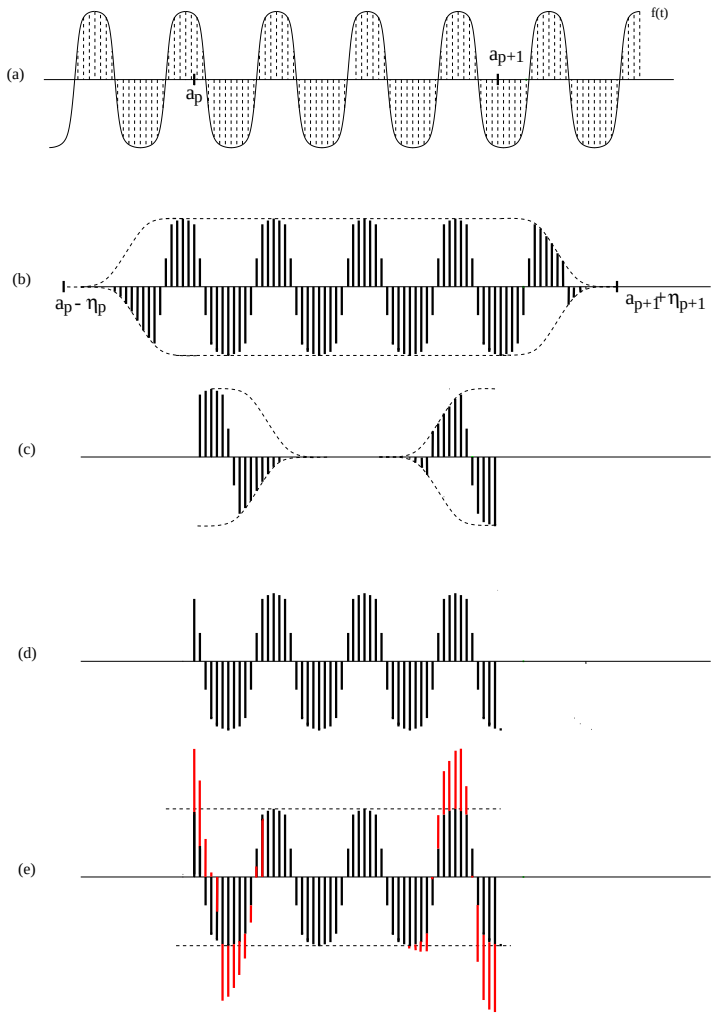


Figura 2.5: Representação gráfica da operação de dobragem. (a) A função $f(t)$, que será dobrada. (b) A versão de $f(t)$ modulada pela função de janelamento. (c) A porção espelhada da função $f(t)$ modulada, que possuía suporte fora do intervalo. (d) A porção de $f(t)$ com suporte no intervalo. (e) A porção espelhada, adicionada à porção de $f(t)$ com suporte no intervalo.

Assim, verificamos que os coeficientes $\langle f, g_{p,k} \rangle$, que representam a transformada do cosseno local de f, podem ser calculados através da dobragem da função f, gerando a função f_{fold}. Em seguida, basta calcular o produto interno da função dobrada com a

base ortogonal do cosseno-IV, o que pode ser feito através da utilização do algoritmo da transformada rápida do cosseno-IV, já descrito.

A transformada inversa do cosseno local reconstrói a função f sobre $[a_p, a_{p+1}]$, a partir dos produtos internos da função dobrada f_{fold} com as funções da base de cossenos-IV. Aplicando a transformada rápida DCT-IV inversa, igual à transformada DCT-IV, reconstruímos a função f_{fold}, a qual, após a aplicação da operação de desdobramento, gera a função f reconstruída.

A operação de desdobramento (*unfolding*) reconstrói $f(t)$ a partir de f_{fold}, através das seguintes fórmulas

$$f(x) = \begin{cases} g_p(x) f_p^{fold}(x) - g_p(2a_p - x) f_{p-1}^{fold}(2a_p - x) & \text{se } x \in O_p^+, \\ f_p^{fold}(x) & \text{se } n \in C_p, \\ g_p(x) f_p^{fold}(x) - g_p(2a_{p+1} - x) f_{p+1}^{fold}(2a_{p+1} - x) & \text{se } x \in O_{p+1}^-, \end{cases} \tag{2.36}$$

onde $O_p^- = [a_p - \eta_p, a_p]$ and $O_p^+ = [a_p, a_p + \eta_p]$.

Como podemos observar na equação 2.36, para reconstruir o sinal f sobre o intervalo $[a_p, a_{p+1}]$, precisamos da função f_{p-1}^{fold} e f_{p+1}^{fold}. Assim, verificamos que é necessário aplicar a DCT-IV inversa nos blocos associados aos intervalos $[a_{p-1}, a_p]$, $[a_p, a_{p+1}]$ e $[a_{p+1}, a_{p+2}]$, antes de realizar a operação de desdobramento. Logo, para a completa reconstrução de um bloco do sinal é necessária a reconstrução parcial de seus vizinhos.

Algoritmo 1 - Cálculo da Transformada Inversa dos Cossenos Locais

```
InverseLCT
{
    for (p=-1; p < Num_intervalos − 1; p++) {
        if (p >= 0 ) {
            DCTIV ( I_p )
            Unfold( I_p )
        }
        DCTIV( I_{p+1} )
        Unfold( I_{p+1} )
        if ( p >= 0 ) {
            AddContrib (I_{p−1}, I_p)
        }
        AddContrib (I_p, I_{p+1})
    }
    AddContrib (I_p, I_{p+1})
}
```

Podemos entender melhor o processo de cálculo da transformada inversa dos cossenos locais, verificando que a mesma é aplicada, para cada intervalo, em três etapas:

1. Aplicação da DCT-IV (DCT-IV);
2. Desdobramento das porções sobrepostas do intervalo (Unfold);
3. Adição da contribuição das porções sobrepostas (AddContrib).

Assim, verificamos que, para a completa reconstrução de um intervalo I_p, é necessária a aplicação das fases 1 e 2 nos intervalos vizinhos I_{p-1} e I_{p+1}. O algoritmo 2.3 apresenta as etapas realizadas para o cálculo da transformada inversa dos cossenos locais.

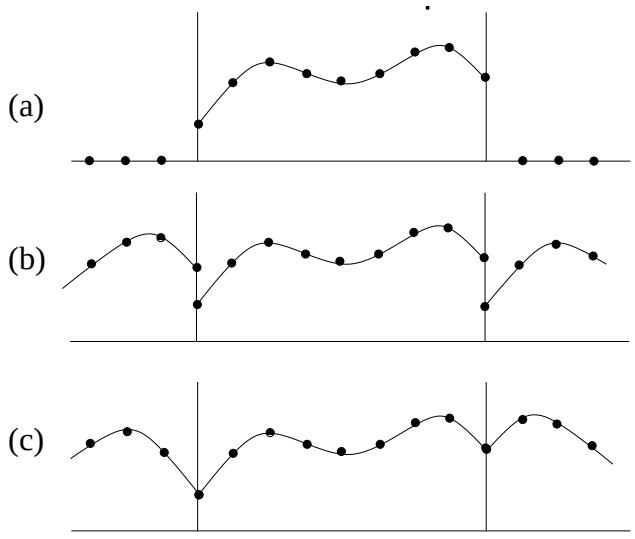


Figura 2.6: Opções para estender o sinal além das fronteiras. (a) Extensão com zeros. (b) Periodização. (c) Reflexão.

Quando o intervalo está na extremidade esquerda ou direita do sinal, não podemos aplicar diretamente o procedimento de dobragem em razão da inexistência de sinal definido na região de sobreposição adjacente. Existem algumas estratégias para estender o sinal nessas regiões de fronteira:

- Extensão do sinal com zeros (Figura 2.6(a)), estratégia utilizada nesta tese;
- Periodização do sinal por translação com $f[N+i] \equiv f[i]$ (Figura 2.6(b));
- Periodização do sinal por reflexão com $f[N+i] \equiv f[N-i+1]$ and $f[-i] \equiv f[i-1]$ (Figura 2.6(c)).

Podemos estender as bases de cossenos locais para representar sinais multi-dimensionais. A maneira mais fácil é através da utilização de produtos tensoriais de elementos de bases undimensionais. Estes produtos tensoriais são ditos separáveis, porque podemos fatorá-los através de somas e integrais para obter uma seqüência de d problemas uni-dimensionais, tratando cada variável de maneira separada.

Baseados nas equações 2.18 e 2.17, podemos verificar que uma base de cossenos locais no espaço 3D pode ser construída particionando o espaço $\mathbb{R}^3$ em intervalos volumétricos $\{[a_p, b_p] \times [c_q, d_q] \times [f_r, h_r]\}$, com comprimento arbitrário $l_p = b_p - a_p$, largura $w_q = d_q - c_q$, e profundidade $s_r = h_r - f_r$. Então, podemos gerar uma família $\{g_{p,q,r,k,i,j}\}_{k,i,j \in \mathbb{Z}^3}$, dada por:

$$g_{p,q,r,i,j,k}(x,y,z) = g_p(x)\, g_q(y)\, g_r(z) \sqrt{\frac{8}{l_p w_p s_p}} \cos\left(\pi i \frac{x - a_p}{l_p}\right) \cos\left(\pi j \frac{y - c_q}{w_q}\right) \cos\left(\pi k \frac{z - f_r}{s_r}\right), \quad (2.37)$$

que representa a base tridimensional de cossenos locais.

2.3.1 Árvore de Cossenos Locais

Na construção das bases de cossenos locais, podemos segmentar o eixo do tempo em intervalos $[a_p, a_{p+1}]$ de tamanho arbitrário. No entanto, Coifman e Meyer (Coifman *et al.* , 1991) mostraram que existem algumas vantagens na restrição dessas partições a intervalos de tamanhos diádicos. Uma dessas vantagens é o fato desta escolha possibilitar a criação de uma estrutura como uma árvore para representar todas as partições diádicas possíveis do eixo do tempo. Uma estrutura desse tipo é denominada de árvore de cossenos locais, sendo muito semelhante à árvore de *wavelet packets* introduzida em (Coifman & Wickerhauser, 1992). Esta árvore pode ser utilizada para a busca das bases de cossenos locais que melhor se adaptam às características do sinal em estudo.

A árvore de cossenos locais é gerada através da segmentação do eixo do tempo em intervalos diádicos. Se considerarmos um intervalo de tempo $[0, T]$ como a extensão do sinal, podemos dividi-lo em $p = 2^j$ intervalos, $I_p = [a_{p,j}, a_{p+1,j}]$, onde $a_{p,j} = p2^{-j}T$ para $0 \leq p \leq 2^j$, que possuem comprimento $l_p = 2^{-j}T$. Os intervalos nos quais $1 \leq p \leq 2^j - 2$ possuem suporte $[a_{p,j} - \eta, a_{p+1,j} + \eta]$, definido pela função de janelamento

$g_{p,j}$ dada por:

$$g_p(t) = \begin{cases} \beta\left(\frac{t-a_{p,j}}{\eta}\right) & \text{se } t \in [a_{p,j}-\eta, a_{p,j}+\eta], \\ 1 & \text{se } t \in [a_{p,j}+\eta, a_{p+1,j}-\eta], \\ \beta\left(\frac{a_{p+1,j}-t}{\eta}\right) & \text{se } t \in [a_{p+1,j}-\eta, a_{p+1,j}+\eta], \\ 0 & \text{de outro modo.} \end{cases} \tag{2.38}$$

Nos intervalos nos quais $p = 0$ ou $p = 2^j - 1$, o suporte é dado por $[a_{p,j}, a_{p+1,j}+\eta]$ e $[a_{p,j}-\eta, a_{p+1,j}]$, respectivamente. Isto é obtido fazendo $g_{0,j}(t) = 1$ se $t \in [0, \eta]$ no primeiro caso, e $g_{2^j-1,j}(t) = 1$ se $t \in [T-\eta, T]$ no segundo caso.

Se calcularmos todos os intervalos diádicos possíveis, geramos uma árvore (mostrada na Figura 2.7), na qual cada nível j representa uma partição do intervalo $[0, T]$, no qual o sinal está definido em 2^j subintervalos. Assim, um nó da árvore de cossenos locais na posição p do nível j é gerado pela família de cossenos locais

$$B_j^p = \left\{ g_{p,j}(t)\sqrt{\frac{2}{2^{-j}T}}\cos\left[\pi\left(k+\frac{1}{2}\right)\frac{t-a_{p,j}}{2^{-j}T}\right]\right\}_{k\in\mathbb{Z}}. \tag{2.39}$$

Para garantir que as janelas se sobrepõem somente com os dois intervalos adjacentes, o comprimento $l_p = a_{p+1,j} - a_{p,j}$ de cada intervalo deve ser maior que $2^{-j}T$, o que resulta do fato de, na partição diádica, o comprimento do intervalo no nível j da árvore de cossenos locais é dado por $l_p = 2^{-j}T$. O valor de η é limitado por: $\eta \leq 2^{-j-1}T$ ou, então, o nível máximo de decomposição é limitado por $\log_2\left(\frac{T}{2\eta}\right)$.

Para representar o sinal, apenas precisamos escolher, entre todos os nós da árvore de cossenos locais, um conjunto de intervalos que preencha completamente o intervalo onde o sinal é definido. Isto pode ser realizado com base na observação de que qualquer intervalo $I_{j,k}$ na árvore pode ser obtido através da soma dos dois intervalos filhos $I_{j+1,2k}$ e $I_{j+1,2k+1}$. Assim, a representação de um sinal pode ser obtida cortando ramos da árvore de cossenos locais e escolhendo somente os intervalos associados às folhas, como ilustrado na Figura 2.8. O número de bases de cossenos locais diferentes é igual ao número de subárvores diferentes com altura de no máximo J.

O tamanho do menor intervalo é limitado pelo tamanho da porção sobreposta (*lap*) $2\eta \leq 2^{-j}N$, o que define a altura máxima da árvore como $J = \log_2 \frac{N}{2\eta}$.

A árvore de cossenos locais se estende para o caso bidimensional em uma *quadtree*, a qual, para cada nó, subdivide as janelas retangulares em quatro subjanelas. Em três dimensões, a estrutura gerada é uma *octree*, que divide o volume em oito subvolumes. Considerando um volume com N^3 *voxels*, temos que um nó na *octree* é

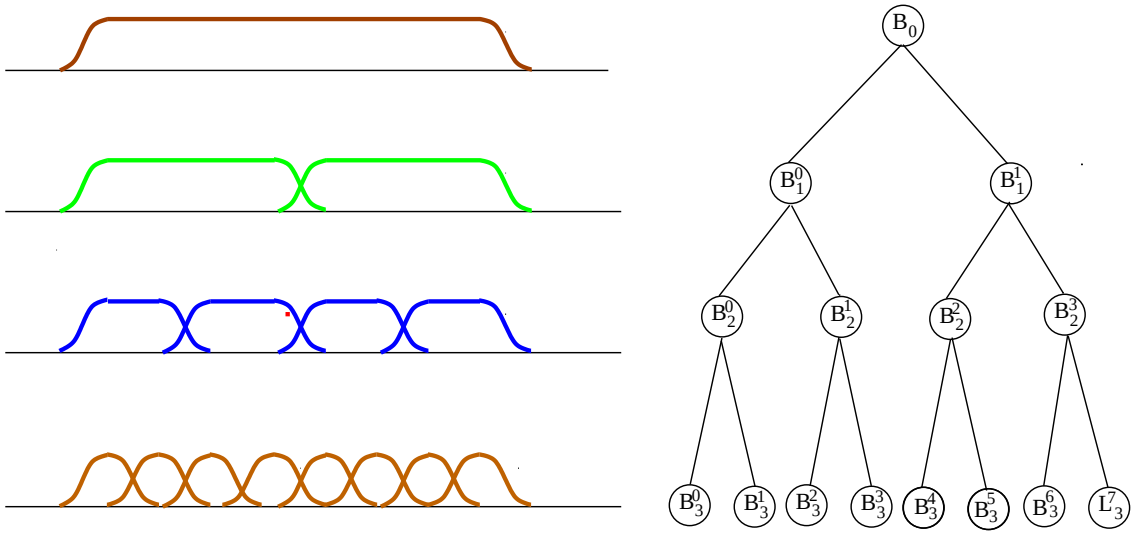


Figura 2.7: Conjunto de intervalos diádicos e a árvore de cossenos locais associada.

definido por sua profundidade j e três índices p, q e r. Na profundidade j, um nó (p,q,r) é gerado por uma base separável de cossenos locais com $2^{-3j}N^3$ vetores

$$B_j^{p,q,r} = \left\{ g_{p,j}[n] g_{q,j}[m] g_{r,j}[l] \sqrt{\frac{2}{2^{-j}N}} \cos\left[\pi\left(k+\frac{1}{2}\right)\frac{n-a_{p,j}}{2^{-j}N}\right] \right.$$
$$\left. \cos\left[\pi\left(i+\frac{1}{2}\right)\frac{m-a_{q,j}}{2^{-j}N}\right] \cos\left[\pi\left(h+\frac{1}{2}\right)\frac{l-a_{r,j}}{2^{-j}N}\right] \right\}. \quad (2.40)$$

A altura máxima da *octree* é definida pelo tamanho da porção sobreposta η, sendo dada por $J = \log_2 \frac{N}{2\eta}$.

2.3.2 Seleção das Melhores Bases de Representação

Bases de cossenos locais dividem o eixo do tempo em intervalos de tamanho variável. Para adaptar a segmentação do tempo na representação do sinal às variações de suas estruturas tempo×freqüência, podemos comparar várias subdivisões obtidas da árvore de cossenos locais para escolher a subdivisão que fornece a melhor representação para o sinal (*best basis*). Minimizando uma função côncava de custos, podemos selecionar, a partir da coleção de representações oferecidas pela árvore de cossenos locais, o subconjunto que é a melhor representação para o sinal. O custo de aproximar um sinal

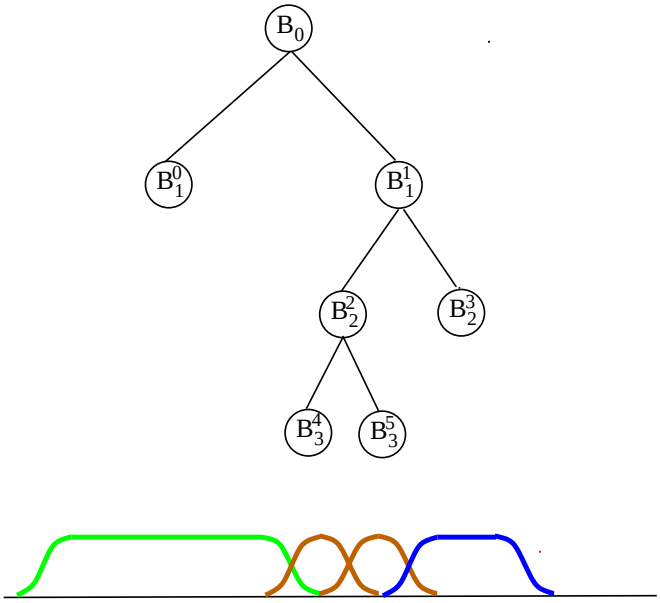


Figura 2.8: Uma árvore de cossenos locais admissível

f em uma base B^λ é definido pela soma côncava de Schür, como:

$$C(f, B^\lambda) = \sum_{m=1}^{N} \Phi\left(\frac{\left\| \left| \left\langle f, g_m^\lambda \right\rangle \right| \right\|^2}{\|f\|^2} \right), \tag{2.41}$$

onde $\left\langle f, g_m^d \right\rangle$ define a representação de f na base B^λ, e Φ é a função côncava de custo.

Como exemplo de funções de custo podemos citar: entropia, concentração em l^p, número de coeficientes acima de um limiar, entropia de um processo de Gauss-Markov, etc.

A entropia, dada por $\Phi(x) = -x \log_e x$, é concava para $x \geq 0$. O custo correspondente é denominado entropia da distribuição de energia, sendo definido por:

$$C(f, B) = -\sum_{m=1}^{N} \frac{\left| \langle f, g_m \rangle \right|^2}{\|f\|^2} \log_e \left(\frac{\left| \langle f, g_m \rangle \right|^2}{\|f\|^2} \right). \tag{2.42}$$

O custo em l^p, $\Phi(x) = x^{\frac{p}{2}}$ para $p < 2$, é concavo para $x \geq 0$. O custo resultante

é dado por:

$$C(f,B) = \sum_{m=1}^{N} \left(\frac{\left| \langle f, g_m \rangle \right|^2}{\|f\|^2} \right)^{\frac{p}{2}} . \tag{2.43}$$

Também podemos definir um valor arbitrário ϵ e contar quantos coefficientes da representação são maiores que ϵ, ou seja, usa a função de custo limiarização dada por: $\phi(x) = \sum \mu(x)$, com:

$$\mu(x) = \begin{cases} x\ , \text{ se } x \geq \epsilon \\ 0, \text{ se } x < \epsilon \end{cases} \tag{2.44}$$

Outra função de custo pode ser usada admitindo que $\phi(x) = \log(x^2)$, então $\sum_{x=1}^{N} \log(x^2)$ pode ser interpretado como a entropia de um processo de Gauss-Markov. Minimizando $\phi(x)$ sobre uma base β, encontra-se a base de Karhunem-Loéve para o processo. E, minimizando $\phi(x)$ sobre uma coleção de bases, encontramos a melhor aproximação da base de Karhunem-Loéve.

Se a coleção de bases é uma árvore com altura finita L, então podemos encontrar o melhor conjunto de bases através do cálculo do custo de representação de cada nó da árvore, e procedendo em seguida uma comparação dos custos dos pais com os dos filhos partindo das folhas em direção à raiz da árvore. A baixa complexidade desse processo é obtida pelo fato de cada nó ser visitado somente duas vezes, o que se torna possível em razão da propriedade aditiva das funções de custo.

O algoritmo para escolha das melhores bases de representação é baseado na premissa de que as bases B_j^p, associadas ao intervalo $I_j^p = I_{j+1}^{2p} \cup I_{j+1}^{2p+1}$, geram um espaço de funções W_j^p, que pode ser gerado através dos espaços (W_j^{2p} e W_{j+1}^{2p+1}) gerados pelas bases (B_{j+1}^{2p} e B_{j+1}^{2p+1}), associadas aos intervalos I_{j+1}^{2p} and I_{j+1}^{2p+1}. Assim, podemos facilmente mudar de uma base B_j^p para as bases B_{j+1}^{2p} e B_{j+1}^{2p+1}. O critério utilizado para selecionar uma das duas representações é a função de custo. O algoritmo se inicia no nível mais subdividido, representado pelas folhas da árvore de cossenos locais, e para cada par de nós compara o custo deles com o custo de seu pai para decidir que representação usar. O processo continua dessa maneira até que o nó raiz seja verificado com seus filhos. A busca em uma decomposição com J-níveis de um sinal com $N = s^J$ amostras necessita de apenas $O(N)$ comparações.

A idéia principal do algoritmo rápido é o corte recursivo em cada nó da árvore, através da comparação da função de custo com a soma dos custos dos filhos correspondentes.

Algoritmo 2 - Escolha da melhor base de representação na árvore de cossenos locais.

$CustoFilhos = \sum_{i=1}^{8} Custo(filho_i)$
if $(Custo(NoPai) \leq CustoFilhos)$ {
    corta os ramos dos filhos.
} else {
    Custo(NoPai) $= \sum_{i=1}^{8} Custo(filho_i)$
}

No início, é produzida uma árvore binária completa, então o processo de corte se inicia a partir das folhas em direção à raiz. No final do processo, uma árvore podada de maneira ótima é obtida para a representação do sinal em questão. Na Figura 2.9, podemos ver uma partição de um volume representando uma possível árvore de *best basis*.

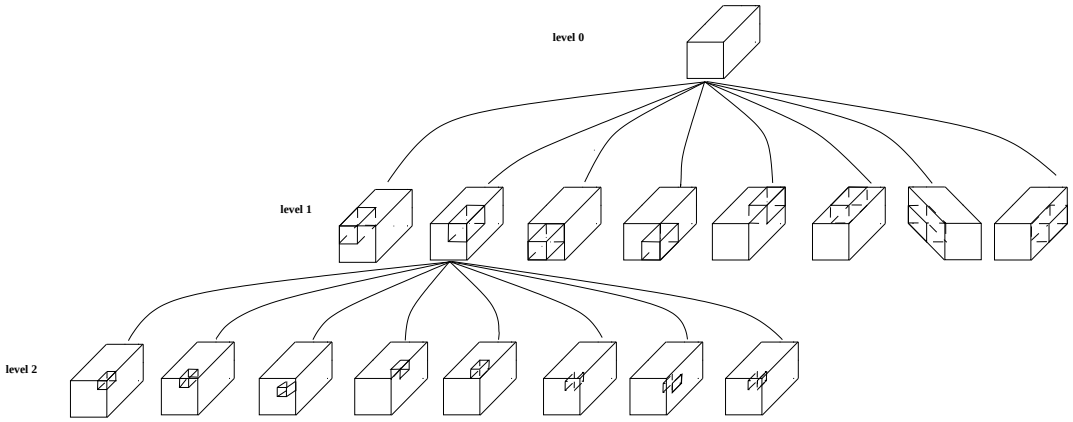


Figura 2.9: Um corte na árvore de cossenos locais de um volume.

Capítulo 3

Compressão de Dados

Compressão de dados é uma prática humana muito antiga, que sempre esteve associada ao objetivo da economia de recursos. Um dos mais antigos exemplos dessa prática são os caracteres Wen-Yan usados pelos chineses para registrar a linguagem falada em um formato compacto, o qual mantêm a essência da informação. Outro exemplo bastante popular é o código de Morse, desenvolvido na métade do século 19 por Samuel Morse. Neste código, as letras são representadas por traços e pontos. Para reduzir o tempo de emissão das mensagens, foram associados códigos mais curtos para as letras que ocorrem com maior freqüência, tais como *e* (.), e códigos maiores para as letras menos freqüentes, como *q* (–.-).

Em geral, os métodos de compressão exploram a estrutura contida nos dados. No caso do código de Morse foi explorada a estrutura estatística dos dados. Diferentes tipos de dados apresentam diferentes estruturas, as quais podem ser exploradas para a obtenção de compressão. Por exemplo, em imagens podemos explorar a coerência espacial dos *pixels* [1]. Além disto, também podemos levar em consideração características dos usuários do dado comprimido. Por exemplo, a compressão de imagens para consumo humano pode levar em consideração as limitações perceptuais de nosso sistema visual.

As técnicas de compressão de dados podem ser divididas, com base nos requisitos de reconstrução, em duas grandes classes: com perda e sem perda.

Os algoritmos de compressão sem perda se caracterizam pela capacidade de reconstrução de uma cópia dos dados idêntica aos dados originais. Shannon demonstrou que essa operação é possível em (Shannon, 1948), e, considerando características estatísticas da fonte de dados, existe um limite teórico para essa compressão, ou seja, o número de bits utilizado para codificar um símbolo com probabilidade p é, no melhor caso, $\log(p)$. O princípio básico desses algoritmos é a diminuição da redundância existente nos dados. Existem vários algoritmos de compressão sem perda, os quais são baseados na estrutura estatística dos dados. Os algoritmos de Huffman, codificação aritmética e Lempel-Ziv estão entre os mais eficientes. Estes métodos são adequados a uma grande variedade de aplicações, embora as taxas de compressão obtidas sejam pequenas tornando-os pouco atrativos. Nesse trabalho nos referiremos a esses métodos

[1] tendência dos *pixels* vizinhos de uma imagem de possuir mesmo valor

como algoritmos de codificação. A Figura 3.1 apresenta um fluxo genérico para algoritmos de compressão sem perda. Para que o algoritmo de codificação funcione de maneira eficiente, é desejável a existência de um modelo preciso para os dados, o que permite que a codificação defina precisamente os tamanhos dos códigos de cada símbolo. Essa separação do método de compressão sem perda em modelagem e codificação foi inicialmente sugerida em (Rissanem & Langdon, 1981).

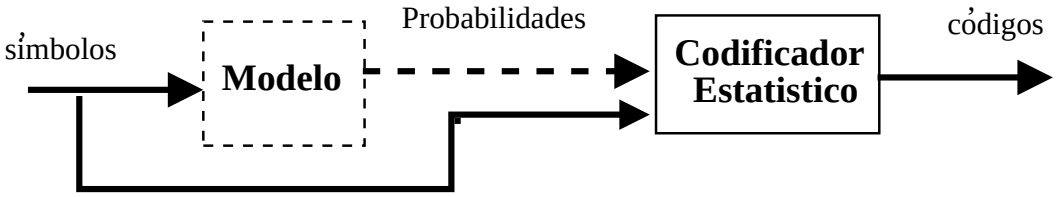


Figura 3.1: Fluxo genérico de algoritmos de compressão sem perdas.

As técnicas de compressão com perda envolvem a perda de parte da informação durante o processo de compressão dos dados, o que resulta na impossibilidade de reconstrução exata do dado original. Em muitas aplicações, as perdas no processo de compressão são aceitáveis. Isto resulta do fato do usuário do dado comprimido não perceber as perdas, ou delas não influenciarem a utilização que se fará do dado reconstruído. Esses algoritmos permitem o ajuste na fidelidade de reconstrução para possibilitar um aumento na taxa de compressão. Esses métodos trabalham no sentido de reduzir parte dos dados que podem ser considerados irrelevantes para uma determinada aplicação. Quando aplicados a sinais, podemos verificar que os métodos de compressão com perda podem trabalhar no sentido de reduzir dois tipos de informação irrelevante:

- Redundância Espacial (Temporal) - correlação entre amostras vizinhas espacialmente (temporalmente) no sinal;
- Redundância Espectral - correlação entre diferentes bandas espectrais.

Quando tratamos de compressão de sinais, temos basicamente os métodos preditivos e baseados em transformadas. Nos métodos preditivos, utiliza-se a informação já codificada para prever valores futuros, e somente a diferença é codificada. Como essa codificação é aplicada no domínio espacial, é relativamente simples de implementar e se adapta de maneira eficiente às características locais do sinal. Um exemplo de algoritmo dessa classe é o DPCM (*Diferential Pulse Code Modulation*). Por outro lado, os métodos baseados em transformadas inicialmente transformam o sinal do domínio espacial para um outro domínio, utilizando uma transformada conhecida, e nesse novo

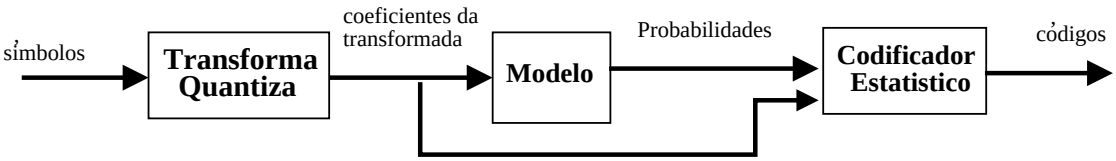


Figura 3.2: Fluxo genérico de algoritmos de compressão baseados em transformadas.

domínio realizam a codificação dos valores transformados (coeficientes). Esses métodos propiciam taxas de compressão maiores que os métodos preditivos, com a desvantagem de necessitarem de um esforço computacional maior. A Figura 3.2 apresenta o fluxo básico de métodos de compressão baseados em transformadas. Esse fluxo é composto por três estágios. O primeiro e mais importante é a transformação da representação do sinal para o domínio mais adequado ao processamento dos estágios seguintes. Em seguida, temos os estágios de quantização e codificação. Essencialmente, verificamos que o estágio inicial determina o que será processado nos estágios subseqüentes.

Nas seções seguintes, concentraremos nossa discussão nas técnicas de compressão de dados que são aplicadas a dados volumétricos. Como dados volumétricos podem ser tratados como uma pilha de imagens, em geral são aplicados a estes dados métodos de compressão de imagem ou extensões para 3D desses métodos.

3.1 Métodos de Compressão sem Perda

3.1.1 Código de Huffman

Código de Huffman (Huffman, 1952) é um dos métodos clássicos de compressão de dados, tendo sido utilizado em uma grande variedade de aplicações. Este método utiliza as propriedades estatísticas dos dados para produzir os códigos para o conjunto de símbolos. Os códigos gerados são de tamanho variável, sendo atribuídos códigos com mais bits para símbolos que possuem menor probabilidade de ocorrência e códigos menores para os símbolos com maior probabilidade. Esta simples idéia causa a redução do tamanho médio do código e, conseqüentemente, o tamanho do dado codificado é menor que o dado original.

Esse algoritmo é baseado na construção de uma árvore binária, que guarda todos os símbolos existentes nos dados em suas folhas, com suas respectivas probabilidades de ocorrência ao lado (Figura 3.3). A árvore é construída seguindo os seguintes passos:

- Cada símbolo presente nos dados forma uma árvore com somente um nó. Na raiz de cada árvore temos a probabilidade de ocorrência da árvore igual à soma das probabilidades das folhas da árvore.

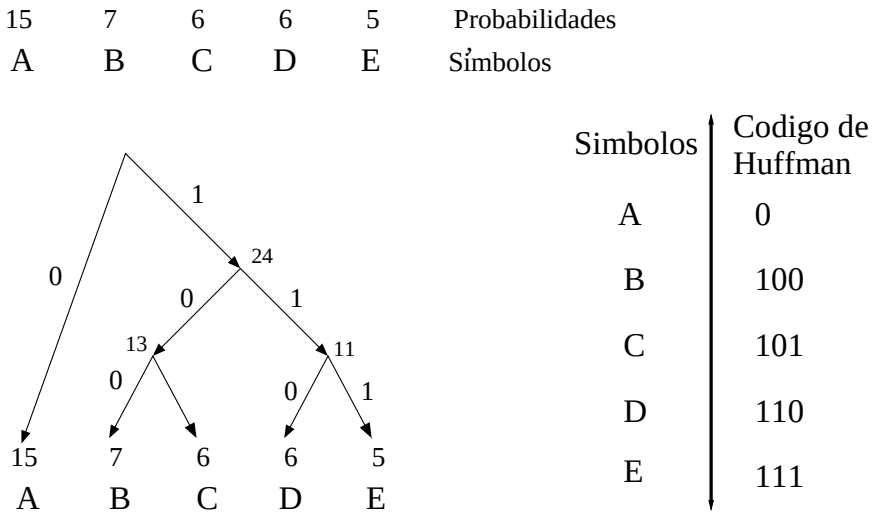


Figura 3.3: Exemplo de codificação utilizando o algoritmo de Huffman.

- As árvores são ordenadas de acordo com a sua probabilidade.
- Duas árvores com as menores probabilidades são combinadas para formar uma nova árvore com um nó raiz como pai.
- As árvores são continuamente reordenadas e combinadas até que uma única árvore binária contenha todos os símbolos dos dados.
- o código representando um símbolo pode ser obtido saindo da raiz da árvore em direção à folha que o contém. Ao percorrer a árvore acrescentamos um 0 ao código quando descemos um ramo para a esquerda, e acrescentamos um 1, quando descemos à direita.

Com a utilização desse procedimento gera-se um conjunto de códigos para os símbolos, cujos comprimentos refletem a sua distribuição de probabilidades de ocorrência. Uma técnica similar para a geração de códigos é o algoritmo de Shannon-Fano, que trabalha de maneira similar ao algoritmo de Huffman, diferindo basicamente na abordagem para a construção da árvore. Shannon-Fano utiliza uma abordagem de cima para baixo, enquanto Huffman usa uma abordagem das folhas para a raiz. Os dois algoritmos apresentam performance bastante similar.

3.1.2 Codificação Aritmética

A idéia de codificação aritmética foi inicialmente mencionada por Shannon (Shannon, 1948), mas a primeira implementação, de acordo com (Abramson, 1963), foi

feita por Peter Elias, que nunca a publicou. O artigo de (Rissanem & Langdon, 1979) é um dos trabalhos que fornecem algoritmos práticos, implementáveis, mais conhecidos. Este método de codificação é baseado na idéia de que é mais eficiente gerar códigos para grupos de seqüências, que gerar um código para cada símbolo em uma seqüência. Como vantagem principal da codificação aritmética podemos citar:

- sua otimalidade teórica, i.e. a geração de códigos para um símbolo com probabilidade p, que possua comprimento muito próximo de $log(p)$, o comprimento de código ótimo de acordo com (Shannon, 1948).
- a grande flexibilidade que resulta da separação entre o codificador e o processo de modelagem (Rissanem & Langdon, 1981).
- adequação aos casos onde existem probabilidade de aparecimento de símbolos muito desbalanceadas, como imagens de dois tons (preto e branco) ou coeficientes de transformadas.

Por outro lado, a codificação aritmética possui as seguintes desvantagens:

- é um algoritmo lento;
- nem sempre gera um código prefixado;
- precisa que se indique com um símbolo especial o final da seqüência;
- possui pouca resistência ao aparecimento de erros.

A codificação aritmética pode ser conceitualmente dividida em duas fases. Na primeira fase é gerado um identificador único (*tag*) para uma dada seqüência de símbolos. Então, na fase subseqüente, é associado um código binário para esse identificador. Outra desvantagem deste método, decorrente da não garantia da geração de um código sem prefixos, é a impossibilidade de sua paralelização.

O algoritmo básico para gerar o identificador para uma seqüência de símbolos se inicia com um intervalo corrente com valor inicial definido para $[0, 1)$. Para cada símbolo na messagem, é associado um subintervalo com tamanho proporcional a sua probabilidade de aparecimento. Então, para cada símbolo na seqüência que está sendo codificada, o intervalo corrente é substituído pelo subintervalo associado ao símbolo corrente, sendo novamente subdividido em intervalos proporcionais às probabilidades dos símbolos. Podemos verificar que o intervalo final gerado para um seqüência particular de símbolos é disjunto de todos os outros intervalos que poderiam ser gerados para outras seqüências de símbolos. Podemos escolher qualquer número dentro desse

Caractere	Probabilidade	Intervalo
		0.0 - 1.0
Espaço	1/10	0.0 - 0.1
A	1/10	0.1 - 0.2
B	1/10	0.2 - 0.3
E	1/10	0.3 - 0.4
G	1/10	0.4 - 0.5
I	1/10	0.5 - 0.6
L	2/10	0.6 - 0.8
S	1/10	0.8 - 0.9
T	1/10	0.9 - 1.0

Tabela 3.1: Exemplo de codificação aritmética. Probabilidades dos símbolos e intervalos associados.

intervalo pra representar a seqüência de símbolos que desejamos codificar. Em geral, escolhemos o limite inferior do intervalo ou o seu ponto médio. Agora, somente precisamos gerar um código binário para esse número.

Para exemplificar, considere a mensagem "BILL GATES" a ser codificada (exemplo extraído de (Nelson, 1991)), a distribuição de probabilidade dos símbolos é dada na Tabela 3.1.

A cada símbolo na mensagem é associado um sub-intervalo do intervalo $[0, 1)$. Assim, verificamos que, para identificar uma mensagem que comece com o símbolo "B", o número que a representa deve estar no intervalo $[0.2, 0.3)$. O próximo símbolo a ser codificado, letra "I", está associado ao sub-intervalo $[0.5, 0.6)$, dentro do novo intervalo corrente $[0.2, 0.3)$. Assim, o número que representa a mensagem "BI" está associado ao sub-intervalo entre 50% e 60% do intervalo corrente, ou seja, um número entre 0.25 e 0.26. Seguindo esse processo, temos os intervalos gerados durante a codificação de todos os símbolos da mensagem representados na Tabela 3.2. Verificamos, pois, que o número 0.2572167752 codifica de maneira única a mensagem.

É relativamente simples verificar como se dá o processo de decodificação. O primeiro símbolo é identificado como aquele cujo intervalo contém o número; no exemplo verificamos que o código está no intervalo entre 0.2 e 0.3, logo, o primeiro símbolo é "B". Então devemos retirar esse símbolo do código. Isto é feito subtraindo o limite inferior do intervalo associado a "B", o que resulta no número 0.0572167752. Então, dividimos o número resultante pelo comprimento do intervalo associado a "B" (0.1), o que resulta em 0.572167752. Para identificar o próximo símbolo, basta verificar em que intervalo esse número cai, no caso símbolo "I". Esse processo continua até ser encontrado um símbolo de marcação de final de mensagem.

Novo Caractere	Limite Inferior	Limite Superior
	0.0	1.0
B	0.2	0.3
I	0.25	0.26
L	0.256	0.258
L	0.2572	0.2576
G	0.25720	0.257220
A	0.257216	0.2572168
T	0.2572164	0.2572168
E	0.25721676	0.257216776
S	0.2572167752	0.2572167756

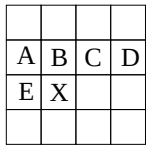
Tabela 3.2: Exemplo de codificação aritmética. Mensagem "BILL GATES".

A implementação básica da codificação aritmética possui dois problemas centrais: a definição do intervalo corrente requer o uso de muita precisão aritmética e, nenhuma saída é gerada até que a completa seqüência de símbolos seja codificada. A solução direta para esse problema é, a medida que conhecemos o valor de um bit que descreve o intervalo, colocarmos esse valor na saída. Podemos decidir que um bit pode ser colocado na saída quando o limite inferior do intervalo possuir o mesmo bit mais significantivo que o limite superior. Após escrever esse bit já conhecido na saída, podemos expandir o intervalo corrente de modo que ele represente somente a porção não conhecida do intervalo final. Existem várias sugestões para lidar com a transmissão incremental de códigos gerados por codificação aritmética e para resolver o problema da precisão aritmética como em (Pasco, 1976), (Rissanem & Langdon, 1979), (Rubin, 1979), (Guazzo, 1980), (Witten & Bell, 1991).

3.1.3 Algoritmos Lempel-Ziv

Codificação LZ é uma técnica de compressão descrita inicialmente em (Ziv & Lempel, 1977) e estendida em (Welch, 1984) para se tornar o algoritmo proprietário LZW. Este algoritmo realiza a compressão através da codificação de cadeias de caracteres. O algoritmo constrói uma tabela com cadeias de caracteres e seus códigos correspondentes, a medida que vai processando o dado.

A primeira vez que é encontrada uma cadeia de caracteres ainda não existente na tabela, essa cadeia é armazenada completamente junto com o código associado a ela. Isto elimina a redundância no arquivo. Para descomprimir, é necessário possuir a seqüência de códigos e a tabela de associação entre as cadeias de caracteres e os códigos.



Estimador Felics

X é estimado usando B e E

Estimador LOCO-I

X é estimado usando A, B e E

O contexto do erro de estimativa é condicionado por A, B, C, D e E

,

Figura 3.4: Estimadores para compressão de imagens.

3.1.4 Codificação por Estimativa

Os métodos de codificação por estimativa têm sido utilizados na compressão de sinais para explorar a correlação entre amostras adjacentes do sinal. Em geral, esses algoritmos estimam o valor de uma amostra do sinal baseados nos valores das amostras vizinhas. Isso resulta numa redução da quantidade de bits necessários para representar o sinal. Alguns métodos de compressão por estimativa aplicados a imagens têm apresentado a melhor performance entre os métodos de codificação sem perda. Bons exemplos são FELICS (*Fast Efficienty Lossless Image Compression*)(Howard & Vitter, 1993) e LOCO-I (*Low Complexity Context-based Lossless Image Compression Algorithm*) (Weinberger *et al.* , 1996). A Figura 3.4 apresenta os esquema de estimativa dos algoritmos citados. Estes métodos são usados em conjunto com outras técnicas de compressão sem perda para codificar os valores de erro dos estimadores. Além disto, existem versões de compressão por estimativas com perda, o que ocorre com a inclusão de uma etapa de quantização dos erros de estimativa. Esses algoritmos de compressão não se apresentam competitivos com os métodos de compressão por transformadas.

3.1.5 Compressão Baseada em Fractais

A idéia de compressão baseada em fractais foi inicialmente proposta para imagens em (Barnsley & Demko, 1985), em razão do princípio básico dos fractais a auto-similaridade. Barnsley observou que alguns objetos naturais podiam ser obtidos como ponto fixo[2] de um certo tipo de função. O problema central a ser resolvido é o seguinte: se uma imagem pode ser obtida como o ponto fixo de um determinado tipo de função, seria possível a partir de uma imagem encontrar a função para a qual a imagem era

[2]Uma função $f(.)$ possui um ponto fixo x_0 se $f(x_0) = x_0$

o seu ponto fixo? Em (Jacquin, 1992) é apresentadia a primeira solução para esse problema.

Assim, é encontrado um conjunto finito de equações matemáticas que descrevem a imagem, sendo necessário codificar somente os parâmetros dessas equações para representar a imagem.

De maneira resumida, verificamos que a compressão fractal é baseada no teorema da contração de mapeamentos da matemática de métricas de espaços. Analisando o sinal, forma-se um sistema de equações denominado PIFS (*Partitioned Iterated Function System*), que é essencialmente um conjunto de contrações de mapeamento. Esses sistemas podem explorar a redundância afim que está comumente presente nos sinais. Essa redundância esta relacionada à similaridade do sinal com ele mesmo, isto é, a parte A de um sinal é similar a uma outra parte B. Aplicando de maneira iterativa o sistema resultante a uma imagem branca, podemos reconstruir completamente a imagem original. Como os sistemas de equações são esparsos, esses métodos conseguem altas taxas de compressão. Entre as desvantagens desse método podemos destacar a baixa eficiência computacional para a compressão e a grande complexidade.

3.2 Métodos de Compressão com Perdas

3.2.1 Compressão Baseada em Transformadas

Esses métodos se baseiam na idéia de que podemos transformar o sinal para um domínio apropriado (representação compacta), descartar os coeficientes que estão próximos de zero, quantizar com poucos bits os coeficientes pequenos e concentrar a representação nos coeficientes que contém a informação mais importante. Dessa maneira, ao reconstruir o objeto, a informação perdida foi a de menor importância.

A figura 3.5 apresenta o fluxo básico de compressão baseada em transformadas para sinais. Este fluxo é composto por três estágios. O primeiro e mais importante é a transformação da representação do objeto gráfico para o domínio mais adequado ao processamento nos estágios seguintes. Em seguida, temos os estágios de quantização e codificação. Na essência, o estágio inicial determina o que especificamente será codificado nos estágios subseqüentes.

Em geral, esses métodos envolvem uma divisão do sinal em blocos que são transformados separadamente, para simplificar o processo de transformação. Porém, essa blocagem pode causar considerável variação no valor reconstruído nas bordas dos blocos.

Figura 3.5: Compressão de sinais.

3.2.1.1 Transformada

O ponto central nesses métodos de compressão é a escolha da transformada utilizada para obter a representação compacta do sinal. Em geral, são usadas transformadas lineares com as seguintes características:

- base de funções fixa;
- base ortogonal - sem redundância;
- algoritmos eficientes para o cálculo;
- separáveis;
- características de redução da correlação entre os coeficientes;
- boas propriedades de compactação da energia;
- boa localização de frequências;
- evita o aprecimento de coeficientes de grande amplitude, resultantes do denominado efeito de borda;

Uma transformada que possui base fixa de funções, um algoritmo eficiente para seu cálculo e boa compactação de energia é a transformada de Fourier. No entanto, a transformada discreta do cosseno (DCT) (Ahmed & Rao, 1974) é uma das transformadas mais usadas para compressão.

A transformada do cosseno local é uma transformada com base ortonormal de classe C^{∞}, suporte compacto e que fornece uma representação em escala para os dados. Adicionalmente, essa transformada possui a vantagem de permitir que a partição do domínio espacial seja feita de maneira adaptativa, baseada nas características do sinal no domínio tempo×freqüência. Em função dessas características, usamos neste trabalho a transformada dos cossenos locais (LCT) descrita no capítulo 2.

Outras transformadas têm sido consideradas para compressão, como Hadamard, discreta do seno e *slant*. Mas, quando a propriedade de compactação de energia é considerada, a DCT apresenta os melhores resultados, pois aproxima a transformada

KL que descorrelaciona os *pixels*. No entanto, estudos recentes mostram que métodos não-lineares de representação podem fornecer resultados melhores que a transformada KL.

3.2.1.2 Quantização

Os coeficientes da transformada são valores reais que precisam ser aproximados por um alfabeto finito e com número pequeno de símbolos, ou seja, precisam ser quantizados. Segundo (Sayood, 1996), podemos definir quantização como o processo que permite representar um conjunto grande de valores, possivelmente infinitos, com um conjunto de símbolos muito menor. No processo de compressão baseado em transformadas, usualmente falamos de quantização escalar, na qual o conjunto dos valores e dos símbolos são formados por escalares.

Esse é o estágio do processo de compressão associado às perdas da informação. Assim, quando executado esse estágio, só é possível reconstruir uma aproximação do dado original. Por outro lado, para compensar essa degeneração nos dados, ganha-se uma maior capacidade de compressão. Em muitas aplicações, a perda resultante da quantização pode ser tolerada, pois não acrescenta distorções visíveis no dado descomprimido.

No processo de quantização, o intervalo de valores dos coeficientes da transformada é divido em subintervalos numerados (*bins*). Qualquer coeficiente com valor dentro de um determinado sub-intervalo é aproximado pelo índice do subintervalo. Em geral, o subintervalo ao redor do zero possui o dobro do tamanho dos outros intervalos. Assim os coeficientes passam a ser representados pelos inteiros no intervalo $[0, 1, \cdots, b-1]$, onde b é o índice associado aos subintervalos.

O processo inverso da quantização é realizado substituindo o índice do subintervalo pelo seu valor médio.

O tipo mais simples de quantização é a quantização uniforme. Nesse caso, os subintervalos de quantização possuem todos o mesmo comprimento. No entanto, é possível variar o comprimento dos subintervalos, de modo que os valores mais populares são quantizados com uma discretização maior (com subintervalos menores). O algoritmo de Lloyd-Max realiza a escolha dos subintervalos de modo a minimizar a distorção introduzida pela quantização.

3.2.1.3 Codificação

Os algoritmos de codificação recaem em duas categorias básicas: métodos baseados em dicionários e métodos estatísticos. Os métodos baseados em dicionários geram

uma representação comprimida do dado através da utilização de códigos de tamanho fixo (em geral 12 ou 16 bits), cada um deles representando uma seqüência particular dos valores originais dos dados. Métodos estatísticos implementam a compressão não-uniforme, representando os valores que mais ocorrem nos dados originais com menos bits. Os métodos de codificação mais usados são:

- Codificação Run Length (RLE).
- Codificação LZW.
- Código de Huffman.
- Codificação Aritmética.

3.2.2 Comparação de Métodos de Compressão com Perdas

Os métodos de compressão baseados em transformadas introduzem erros e distorções, sendo, portanto, denominados métodos de compressão com perdas. No entanto, métodos diferentes produzem tipos e quantidades diferentes de erros e distorções. Mesmo quando os erros introduzidos podem ser quantificados, sua magnitude pode não descrever adequadamente a influência visual desses erros introduzidos. Assim, verificamos que, em última instância, é necessário que a avaliação do método de compressão seja realizada através de experimentos envolvendo o usuário dos dados. Isso significa, por exemplo, avaliar os métodos de compressão de dados volumétricos através da avaliação subjetiva de um grande número de observadores sobre imagens geradas a partir do volume reconstruído. Ou, no caso de dados sísmicos, verificar a influência da perda em processamentos do dado volumétrico, tais como a migração.

No entanto, existem critérios objetivos que podem ser utilizados para comparar diversos métodos de compressão. Em geral, esses critérios buscam definir a quantidade de compressão conseguida e quão próximo do original é o dado reconstruído.

Para verificar quão bem um método de compressão comprime um conjunto de dados, basta verificar a razão entre o número de bits necessários para representar os dados antes e após a compressão. Essa razão é denominada razão de compressão (CR). Outra medida é a razão que define o número de bits por unidade de informação. Para imagens definimos o número de bits por *pixel* (bpp) e para volumes utilizamos bits por *voxel*.

Para quantificar a qualidade da reconstrução, é necessário verificar a diferença entre os dados originais e os dados reconstruídos, denominada distorção. Duas populares medidas de distorção são o erro quadrático e a diferença absoluta. Considerando

que $\{x_n\}$ representa o dado original e $\{y_n\}$ representa o dado reconstruído após a compressão, então temos que o erro quadrático é dado por $se(x, y) = (x - y)^2$, e a diferença absoluta é dada por $d(x, y) = \|x - y\|$.

Por causa da dificuldade de analisar os erros obtidos em cada amostra do dado, é mais comum trabalharmos com medidas médias. A mais usada é a média do erro quadrático, denominada de erro médio quadrático (MSE), dada por:

$$\sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n - y_n)^2 \tag{3.1}$$

Podemos, também, analisar o tamanho do erro em relação à magnitude do sinal comprimido, o que é fornecido pela razão sinal ruído (SNR), que pode ser aproximada por:

$$SNR = \frac{\sigma_x^2}{\sigma^2}, \tag{3.2}$$

onde σ_x^2 é a média dos quadrados dos valores do sinal original e σ^2 é o erro médio quadrático. Para apresentar os valores de SNR em decibéis (dB) o representamos em escala logarítmica. Assim, $SNR_{dB} = 10\log_{10}(SNR)$.

Se estivermos mais interessados em quantificar o erro de maneira relativa à maior amplitude do sinal x_{max}, utilizamos a razão sinal ruído de pico (PSNR) dada por:

$$PSNR_{dB} = 10\log_{10}\frac{x_{max}^2}{\sigma^2}, \tag{3.3}$$

onde x_{max} é definido como o valor da amostra de maior amplitude.

Podemos, então, comparar os métodos através de suas curvas de razão de distorção, analisando as respostas geradas para uma determinada razão de compressão. Um método apresenta resultado melhor que outro se suas curvas de razão de distorção apresentam na média ou de maneira consistente menor distorção que o outro.

Para os métodos baseados em transformadas uma boa métrica é a utilização de uma medida da eficiência da transformada em concentrar a energia dos dados originais em um pequeno número de coeficientes. Essa medida é denominada de ganho da transformada (Jayant & Noll, 1984)(*Transform Coding Gain* - TCG), sendo calculada como a razão entre a média aritmética da variância dos coeficientes da transformada e

a média geométrica dessas variâncias, como representado em:

$$TCG = \frac{\frac{1}{N}\sum_{i=0}^{N-1}\pi^2}{\left(\prod_{i=0}^{N}\pi^2\right)^{\frac{1}{N}}}, \tag{3.4}$$

onde π^2 é a variância do i-ésimo coeficiente da transformada. Em (Wickerhauser, 1993), é apresentado um experimento mostrando que o ganho de transformada da LCT é maior que o da DCT, fornecendo uma melhor aproximação da transformada de Karhunem-Loéve.

O ganho da transformada não explicita em que faixa de freqüência a compactação de energia ocorreu. Para verificar a energia concentrada nos primeiros m coeficientes, podemos utilizar a eficiência da transformada (*Transform Efficiency - TEF*). A eficiência da transformada é uma medida da energia mantida em um número particular de coeficientes: medimos a energia mantida nos m primeiros coeficientes e dividimos pela energia mantida em todos os coeficientes, multiplicando por 100 para representá-la em percentual. Isso fornece uma indicação da energia mantida nos coeficientes de baixa freqüência.

Capítulo 4

Compressão Volumétrica Baseada na Transformada do Cosseno Local.

4.1 Esquema de Compressão Proposto.

A figura 4.1 descreve o fluxo geral do método de compressão proposto, o qual segue um fluxo genérico de métodos baseados em transformação. O estágio inicial é a aplicação da transformada do cosseno local, gerando o volume transformado. Em seguida, temos dois estágios típicos de um codificador com perdas: quantização dos coeficientes restantes para restringi-los a um pequeno número de possibilidades; e, finalmente, a codificação dos dados transformados.

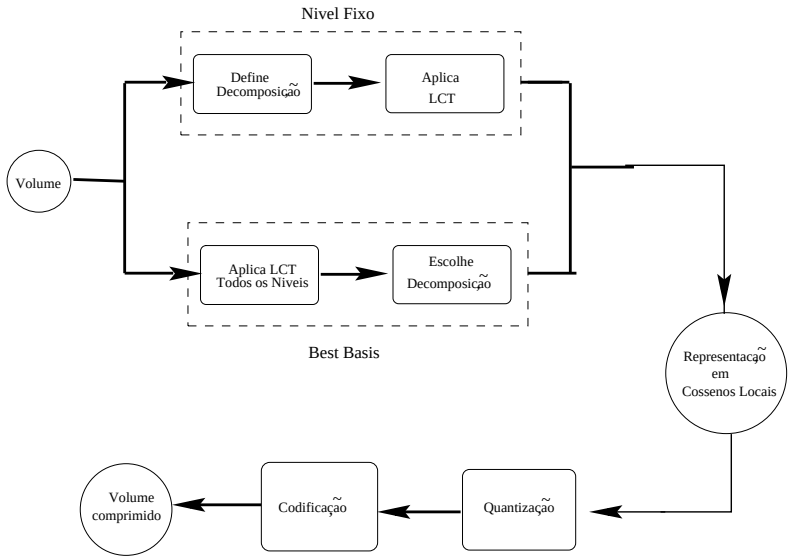


Figura 4.1: Visão geral do método de compressão de volumes baseado na transformada do cosseno local

4.1.1 Transformação do Volume

O estágio inicial do método de compressão é a definição da partição do volume para aplicação da transformada do cosseno local. Essa partição pode ser definida usando o método de nível de decomposição fixo, o qual subdivide o volume em subvolumes de tamanhos iguais ($n_x \times n_y \times n_z$), ou usando o método baseado em decomposição

adaptativa que utiliza o algoritmo de *best basis* para encontrar de maneira adaptativa a melhor subdivisão para representação do volume. Nesse caso, temos que a descrição da decomposição do volume deve ser codificada de uma maneira que seja possível ao método de descompressão recuperar o arranjo utilizado para os blocos da decomposição. É desejável que a codificação do arranjo dos blocos permita a sua recuperação em qualquer ordem. Esta característica é fundamental para tornar possível o desenvolvimento de algoritmos de visualização baseados na estrutura dos dados comprimidos, os quais descomprimem o volume localmente, minimizando os requisitos de memória para a visualização.

Usando a abordagem de método de decomposição fixo, especificamos o nível de decomposição diádica que será utilizado e o tamanho da porção sobreposta dos intervalos (*lap*). Com base nessas definições, o algoritmo calcula a partição do volume. Neste trabalho utilizamos apenas as partições diádicas do volume ao longo das três direções principais. Assim, um volume de tamanho $N_x \times N_y \times N_z$ possui 2^{3j} blocos no nível j de decomposição diádica, cada bloco com tamanho $\frac{N_x}{2^j} \times \frac{N_y}{2^j} \times \frac{N_z}{2^j}$. O nível máximo de decomposição diádica j_{max} é determinado pelo tamanho da porção sobreposta das janelas de cossenos locais (*lap*), sendo calculado através da expressão $j_{max} = \log_2(2 * lap)$. Na transformação implementada para a realização desse trabalho, utilizamos o mesmo tamanho para o intervalo de sobreposição em todos os blocos.

Para cada bloco resultante da decomposição do volume, aplicamos três transformadas de cossenos locais unidimensionais, uma ao longo de cada eixo, adicionando as contribuições de dobragem dos seis vizinhos. Esse processo pode ser implementado com a aplicação da transformada dos cossenos locais unidimensional a $N_x \times N_y$ vetores unidimensionais formados pelos *voxels* (i, j) de cada fatia do volume, usando 2^j intervalos para cada transformada. Segue-se a aplicação da transformada unidimensional com 2^j intervalos às colunas de cada fatia resultante da transformada anterior, seguido do mesmo processo aplicado às linhas de cada fatia resultante do processo precedente. Para o cálculo da transformada dos cossenos locais de um volume é necessária apenas uma memória adicional de N_{max} posições para armazenar os valores dos *voxels*, que correspondem aos vetores unidimensionais montados a partir das posições (i, j) de cada fatia, e aos que correspondem às colunas das fatias já transformadas na direção z. O valor de N_{max} é definido como o maior valor entre N_y e N_z.

A segunda abordagem é baseada na busca por uma partição do volume que seja adaptada às características de freqüência do volume. Nesse método, é definido um valor para o tamanho do intervalo de sobreposição, o qual define o nível máximo de decomposição que pode ser utilizado. Em seguida, as transformadas do volume são armazenadas em um volume de blocos, cada bloco sendo um nó da árvore de

cossenos locais. O processo é repetido para o nível de decomposição anterior. Assim temos dois volumes de nós da árvore de cossenos locais, um representando um nível de decomposição L e o outro o nível $L-1$. Então, para cada nó (b_i^L, b_j^L, b_k^L) do volume de blocos do nível $L-1$, calculamos o custo da representação e comparamos com seus oito filhos no nível L, os blocos:

$$(b_{son_k}, b_{son_i}, b_{son_j}) \quad (b_{son_k}, b_{son_{i+1}}, b_{son_j}) \quad (b_{son_k}, b_{son_i}, b_{son_{j+1}}) \quad (b_{son_k}, b_{son_{i+1}}, b_{son_{j+1}})$$

$$(b_{son_k}, b_{son_i}, b_{son_j}) \quad (b_{son_k}, b_{son_{i+1}}, b_{son_j}) \quad (b_{son_k}, b_{son_i}, b_{son_{j+1}}) \quad (b_{son_k}, b_{son_{i+1}}, b_{son_{j+1}}),$$

onde $son_i = 2 * i$, $son_j = 2 * j$ e $son_k = 2 * k$.

Caso o somatório dos custos dos filhos seja menor que o custo do pai, então escolhemos a representação do volume que está associada aos nós da árvore de cossenos locais associados aos filhos. Isto significa retirar da árvore de cossenos locais o bloco de transformada associado ao nó pai, e substituir seu custo pelo somatório dos custos dos filhos. Caso contrário os filhos são descartados, tornando o nó pai uma folha da árvore. Ao final do processo temos uma *octree* recortada, com os nós que foram escolhidos para representar o volume localizados em suas folhas. Para o processo de compressão, realiza-se um caminhamento em profundidade na *octree*, para cada nó da árvore descemos nas sub-árvores segundo uma ordem fixa, primeiro a fatia de blocos zero, em cada fatia, primeiro a linha zero e, em cada linha, primeiro a coluna zero até a última coluna. Em cada folha da árvore, o bloco de transformada associado é enviado para o processo de quantização/codificação. Em essência, esse é o funcionamento do algoritmo de compressão de volumes baseado em *best basis*.

O custo de adicionar adaptatividade ao esquema de compressão é um cabeçalho descrevendo a decomposição do volume em blocos. Por exemplo, para um volume de tamanho 256^3, e tamanho da porção sobreposta maior que 1, podemos ter no máximo 64 blocos em cada direção. Assim, com um cabeçalho de 18 bits para cada bloco podemos identificar a posição do bloco em seu nível de decomposição, com mais 3 bits identificamos o nível. Logo, no máximo precisaremos de 21 bits por bloco para codificar seu nível e posição. Verificamos então, que o esquema adaptativo necessita de no máximo 5376 bytes, Considerando que temos no máximo 262144 (64^3) blocos, então o esquema adaptativo necessita, na pior hipótese, de 5376 bytes adicionais, o que para um volume com 1 byte por *voxel* representa um acréscimo de 0.00032 bits por *voxel*. É necessário, portanto, avaliar quando essa memória adicional é compensada pelo ganho de compressão obtido pela utilização da base adaptativa. No esquema de compressão proposto, armazenamos, para cada bloco da representação em *best basis*, quatro valores inteiros que serão codificados junto com os coeficientes quantizados.

Esses valores representam o nível de decomposição e a posição (b_i, b_j, b_k) desse bloco no nível em que está localizado.

4.1.2 Quantização e Codificação

O processo de quantização utilizado foi uniforme. Assim, foi definido um passo de quantização constante para cada bloco do volume, o qual foi calculado de maneira semelhante à proposta em (Chiueh *et al.* , 1997). Para todos os elementos em um bloco, calculamos o intervalo dinâmico dos coeficientes $R_{bi,bj,bk}$ através da expressão

$$R_{bi,bj,bk} = \|maxValue_{bi,bj,bk} - minValue_{bi,bj,bk}\|, \tag{4.1}$$

onde $maxValue_{bi,bj,bk}$ e $minValue_{bi,bj,bk}$ são os valores máximo e mínimo dos coeficientes da transformada para um dado bloco (bi, bj, bk). O passo de quantização para um subvolume é calculado, então, por

$$Q_{bi,bj,bk} = \frac{R_{bi,bj,bk}}{2^c}, \tag{4.2}$$

com c definido como uma constante fixa. Podemos definir a constante c como o número de bits que usaremos para representar os coeficientes quantizados da transformada antes da codificação. Nesta tese, utilizamos os valores de 8, 10 e 12 para a constante c.

Cada um dos coeficientes é então quantizado utilizando a equação:

$$F^Q(i,j,k) = \lfloor \frac{F(i,j,k)}{Q_{bi,bj,bk}} + 0.5 \rfloor - F_{min}, \tag{4.3}$$

ou seja, o valor do coeficiente quantizado é resultante do arredondamento da razão entre o coeficiente original e o passo de quantização correspondente ao bloco em que se encontrai, além de uma translação para colocar o coeficiente de menor valor no bloco (F_{min}) na origem.

Após a quantização, os coeficientes sofrem uma translação igual ao valor do menor coeficiente quantizado do bloco, isto é necessário para obtermos todos os coeficientes como valores positivos, o que torna o algoritmo de codificação aritmética mais eficiente. Assim, para cada bloco do volume é também codificado o passo de quantização utilizado e o valor do menor coeficiente quantizado. O valor do passo de quantização é armazenado sem codificação, enquanto o coeficiente mínimo é codificado junto com os demais coeficientes.

A compressão se dá, além da quantização, através da manutenção de apenas um pequeno número de coeficientes de cada bloco. Para selecionar os coeficientes

mais significantes seguimos a abordagem de Chen e Pratt (Chen & Pratt, 1984), que sugeriram que o caminhamento ao longo do bloco ocorresse segundo uma ordem em zigzag. A ordem em zigzag tridimensional é o caminhamento ao longo de um bloco de modo que (u_1, v_1, w_1) é visitado antes de (u_2, v_2, w_2) se $u_1 + v_1 + w_1 < u_2 + v_2 + w_2$; as tuplas (u, v, w) contidas no plano $u + v + w = K$ são percorridas segundo uma ordem em zigzag bidimensional. A Figura 4.1.2 ilustra essa ordem de caminhamento ao longo dos blocos. Essa abordagem é baseada na suposição que um grande número dos coeficientes encontrados por último, segundo essa ordem, será formada basicamente por zeros. Isso porque, em geral, os coeficientes de alta freqüência possuem menor amplitude. Para verificar essa suposição, calculamos a eficiência da transformada para vários dados utilizando o caminhamento ao longo do bloco na forma convencional (fatia por fatia, em cada fatia caminhamento linha por linha) e na forma de zigzag. Neste experimento pudemos verificar que, quando percorrido em zigzag, a eficiência cresce consideravelmente, conforme será mostrado na seção seguinte junto com os demais resultados.

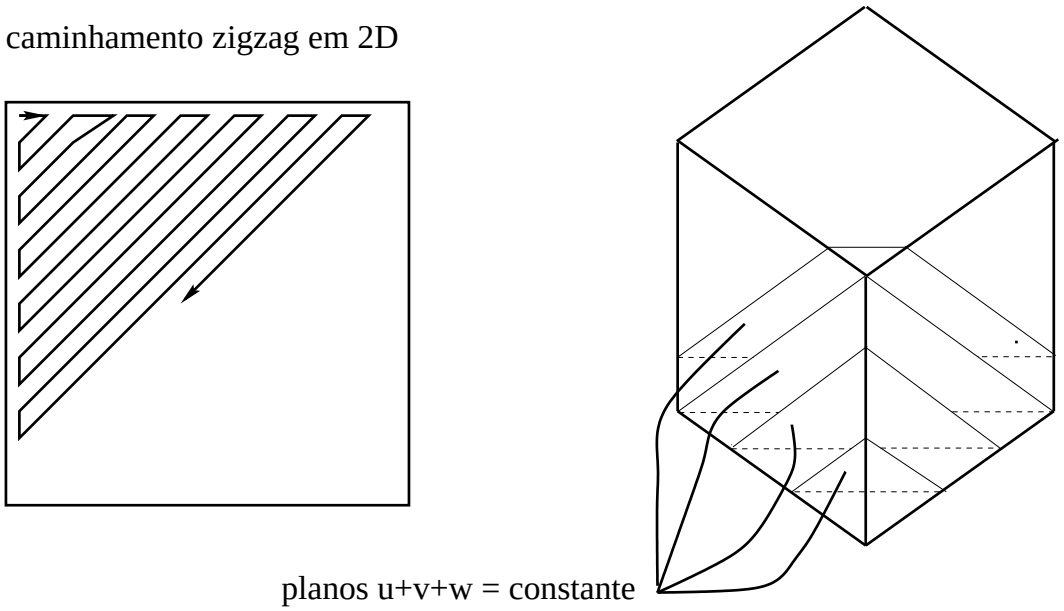


Figura 4.2: Ilustração do caminhamento em zigzag em um bloco do volume.

A sequência linear de coeficientes resultante do caminhamento em zigzag dos blocos da transformada é submetida a um corte, sendo mantidos apenas uma porcentagem dos coeficientes de mais baixa freqüência, definindo implicitamente os demais como zero. Os coeficientes são, então, enviados para um processo de geração de códigos com baixa entropia por codificação aritmética. A codificação aritmética utilizada é aplicada sobre todos os coeficientes retidos do volume transformado. Há uma primeira etapa em

que um modelo estatístico de primeira ordem estima as freqüência de cada coeficiente, sendo essa etapa seguida pela geração dos códigos de cada coeficiente. Ao final de cada bloco é colocado um valor adicional para determinar fim de bloco.

A implementação de codificação aritmética utilizada nesse trabalho foi desenvolvida e disponibilizada por Alistair Moffat, tendo sido baseada nos trabalhos (Moffat *et al.* , 1998) e (Moffat, 1999). Nos testes realizados foi utilizado o modelo no qual os coeficientes que serão codificados são representados em 2 bytes sem sinal.

A descompressão é obtida através da aplicação na ordem inversa das etapas de compressão. Inicialmente são decodificados os códigos de baixa entropia, gerando os valores dos coeficientes. Em seguida, os demais coeficientes do bloco são definidos como valor zero. Aplica-se, então, a transformada LCT inversa.

Inicialmente é necessário decodificar os dados comprimidos e aplicar a translação inversa dos coeficientes, subtraindo o valor absoluto do coeficiente mínimo. Em seguida, os coeficientes sofrem a operação inversa da quantização sendo calculados pela expressão:

$$F(i,j,k) = F^{Q}(i,j,k) * Q_{bi,bj,bk}. \tag{4.4}$$

Para permitir a visualização de um bloco da transformada por vez, foi desenvolvido um algoritmo para a reconstrução local de volumes comprimidos usando a transformada do cosseno local.

Para descomprimir um bloco na posição (b_i, b_j, b_k), é necessário aplicar a DCT-IV e a operação de desdobramento nos seus seis vizinhos. Como a transformada do cosseno local 3D e sua inversa são aplicadas através do cálculo de três transformadas unidimensionais, temos que realizar esse processo ao longo das três direções. Inicialmente é aplicado a DCT-IV e o desdobramento ao longo da profundidade do volume no blocos (b_i, b_j), (b_i, b_{j+1}), (b_{i+1}, b_j) e (b_{i+1}, b_{j+1}) das fatias k e $k+1$. Em seguida são adicionadas as contribuições entre os blocos adjacentes na direção z. Então, aplicamos a DCT-IV e o desdobramento ao longo da direção y nos blocos (b_i, b_j), (b_{i+1}, b_j), (b_i, b_{j+1}) e (b_{i+1}, b_{j+1}) da fatia k, adicionando as contribuições entre os blocos adjacentes ao longo de y. Finalmente, é aplicada a DCT-IV e o desdobramento ao longo de x nos blocos (b_i, b_j) e (b_i, b_{j+1}) da fatia k, e, logo após, é adicionada as contribuições entre esses blocos. Assim, o bloco (b_i, b_j, b_k) está completamente reconstruído podendo ser, por exemplo, enviado para um processo de visualização. Além disto, outros blocos envolvidos na reconstrução do bloco (b_i, b_j, b_k) estão parcialmente reconstruídos. Podemos verificari, para exemplificar, que o bloco (b_{i+1}, b_j, b_k) já foi submetido a duas transformadas de cossenos locais unidimensionais, faltando para sua completa reconstrução apenas a aplicação da transformada ao longo de x.

Dado	Número de *Voxels*	bits/*voxel*	Tipo	Tamanho (Kbytes)
3DHead	256×256×64	8	MRI	4096
Colt	256×256×64	8	SEISMIC	4096
Broncho	512×512×128	8	CT	8192
VH	512×512×128	16	CT	65536

Tabela 4.1: Descrição dos dados volumétricos usados para testes.

O cálculo da transformada inversa do cosseno necessita, portanto, de memória para manter os blocos parcialmente reconstruídos. No pior caso essa memória é equivalente a duas fatias de blocos, ou seja, duas vezes a resolução das fatias do volume vezes o número de fatias em cada bloco.

A implementação deste algoritmo para a representação em *best basis* é mais complexa, em razão das adjacências dos blocos serem irregulares.

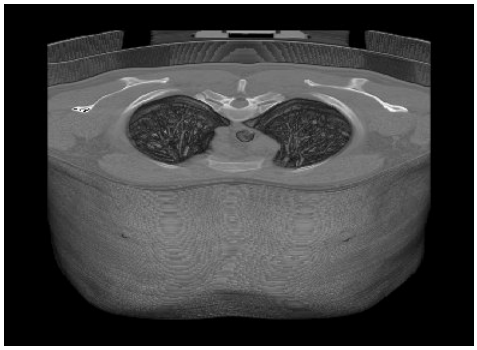
4.2 Resultados Obtidos

4.2.1 Dados Utilizados para Testes

Para avaliar os resultados obtidos com o esquema de compressão proposto e a influência dos parâmetros da transformada na compressão e no erro de reconstrução, utilizamos 4 conjuntos de dados. Esses dados podem ser divididos em três categorias: ressonância magnética (MRI), tomografia computadorizada (CT) e levantamento sísmico. Assim, cobrimos três importantes fontes geradoras de dados volumétricos. Além disto, nos dados de CT e Sísmica, temos volumes com os valores dos *voxels* previamente quantizados para 8 bits e com sua representação original. Particularmente para dados sísmicos, a quantização para 8 bits é feita somente com fins de visualização. As principais características dos volumes usados nos testes estão descritas na Tabela 4.1.

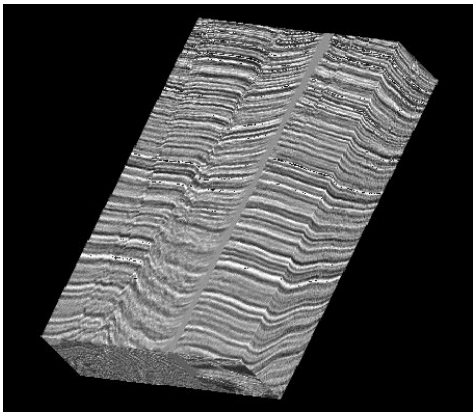
Os dados 3DHEAD e COLT foram obtidos do CD que acompanha o livro (Barthold *et al.* , 1997). O dado 3DHEAD é uma ressonância magnética de uma cabeça, com o crânio parcialmente removido de modo a revelar o cérebro. Seu tamanho original é 256×256×109 *voxels*, tendo sido diminuido para 256×256×64, em razão da transformada implementada ser aplicável somente a volumes com dimensões potência de dois. O corte dos volumes foi realizado simplesmente descartado os voxels fora do novo tamanho especificado. Os dados foram quantizados para 8 bits por *voxel*, a partir de sua representação original de 16 bits. Esse dado é disponibilizado pela Siemens Medical Systems, Inc., Iselin, NJ.

(a) 3DHEAD

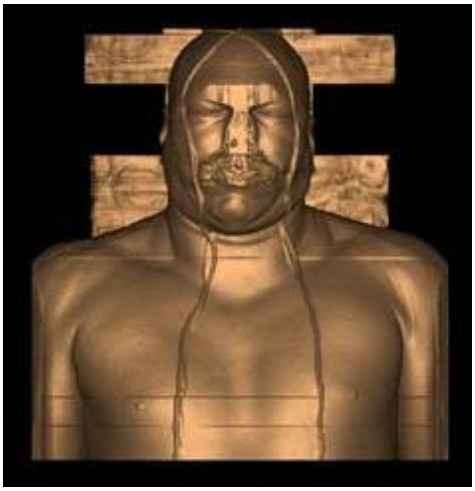


(b) BRONCHO

Figura 4.3: Dados utilizados para teste.



(a) COLT



(b) VH

Figura 4.4: Dados utilizados para teste.

O volume COLT originalmente com 300×455×91 *voxels*, cada um deles armazenado em 8 bits, representa um dado sísmico adquirido por um levantamento em mar ou terra. Para ser usado nesta tese, o volume foi reduzido para o tamanho de 256×256×64 *voxels*. Este volume é disponibilizado como uma cortesia de Landmark Graphics Corp.

O dado BRONCHO é uma tomografia computadorizada da região do tronco de um paciente, tendo sido disponibilizado para uso no Tecgraf por um de seus parceiros médicos.

O volume VH é o mesmo dado utilizado em (Ihm & Park, 1998), (Rodler, 1999) e (Kim & Shin, 1999). Esse dado foi disponibilizado pelo Professor Insung Ihm. O volume utilizado é uma parte do volume original com $512 \times 512 \times 128$ *voxels*, reamostrado a partir dos dados de CT adquiridos com o cadáver fresco[1]. Esta reamostragem se fez necessária em razão da existência de diferentes espaçamentos entre fatias e tamanhos de *pixels* nos dados originais. Os valores dos *voxels* variam no intervalo $[0, 4095]$ (12 bits), estando armazenados em 16 bits.

4.2.2 Resultados da Transformada

Para analisar o comportamento da transformada 3D, foi realizada uma série de testes, calculando a eficiência da transformada (TEF) para cada um dos dados, de modo a verificar a influência da discretização do espaço e do tamanho do intervalo de sobreposição sobre a concentração de energia.

Para o dado 3DHEAD, foram analisados os efeitos da ordenação dos coeficientes segundo o caminhamento em zigzag ao longo dos blocos com o caminhamento normal e segundo a ordem das fatias. O gráfico da Figura 4.5 mostra um ganho de 10 a 25% na TEF para o caminhamento em zigzag nos níveis de menor retenção de coeficientes. Podemos observar que o ganho é maior nos níveis em que os blocos de transformada são maiores.

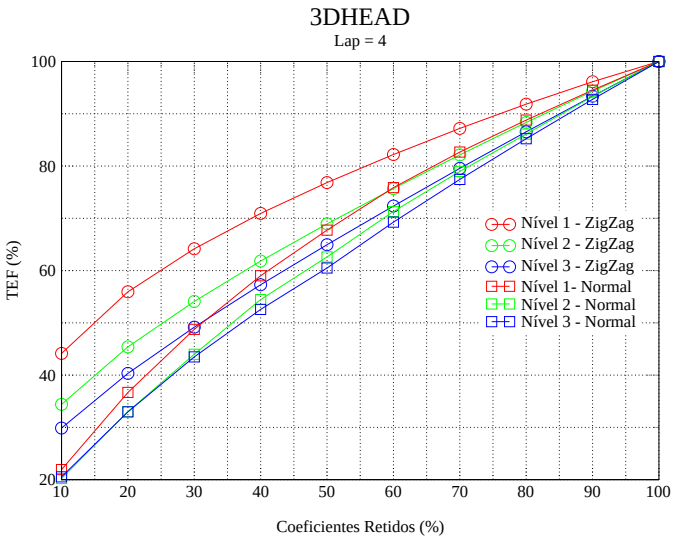


Figura 4.5: Eficiência da transformada com caminhamento normal e em zigzag - Volume 3DHEAD.

[1]Existem dados de CT do Visible Human com o cadáver congelado

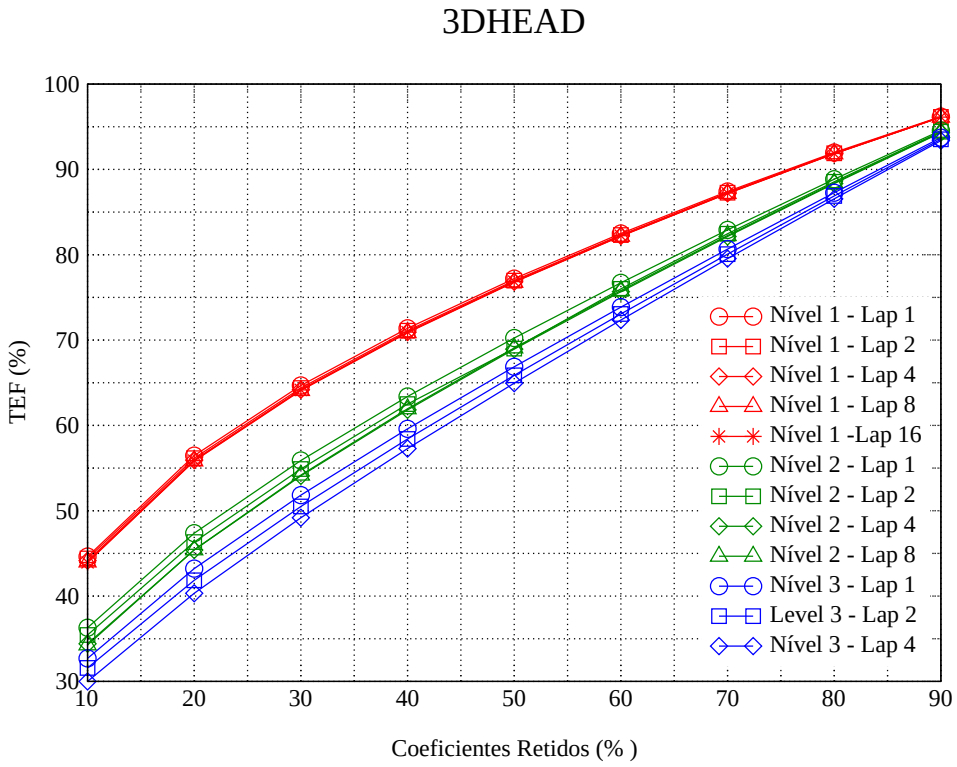


Figura 4.6: Influência do tamanho do intervalo de sobreposição (*lap*) sobre a TEF - Volume 3DHEAD.

Também foi analisada a influência do tamanho do intervalo de sobreposição sobre a concentração de energia, conforme mostrado no gráfico da figura 4.6, no qual todas as curvas correspondem ao caminhamento em zigzag. Nesse gráfico verificamos que o tamanho do intervalo de sobreposição passa a influênciar a concentração de energia nos primeiros coeficientes a medida em que aumenta a razão entre o seu tamanho e o tamanho do intervalo de aplicação da transformada. Assim, verificamos que, para o nível 1, o volume 3DHEAD gera blocos com 128×128×32 coeficientes, o *lap* praticamente não influencia a concentração de energia. Já para o nível três, no qual temos 32×32×16 coeficientes em cada bloco, cada vez que dobramos o *lap* diminuímos a TEF em 1%.

A Figura 4.7 mostra as curvas que representam a eficiência da transformada calculada para o volume BRONCHO. Para esse volume, o caminhamento em zigzag mostrou-se ainda mais eficiente. Podemos verificar que o ganho em relação ao caminhamento normal chega a mais de 35% na TEF. Novamente temos que os níveis de menor decomposição apresentam uma menor concentração de energia.

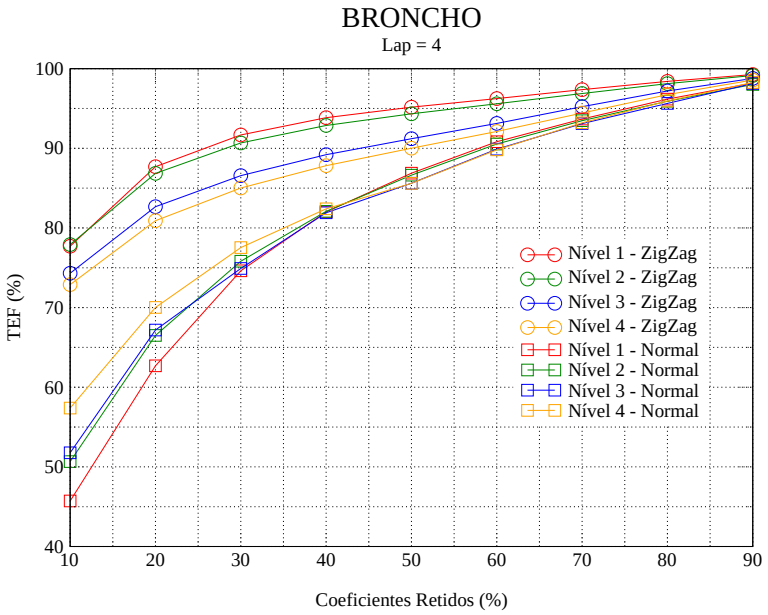


Figura 4.7: Eficiência da transformada com caminhamento normal e em zigzag - Volume BRONCHO.

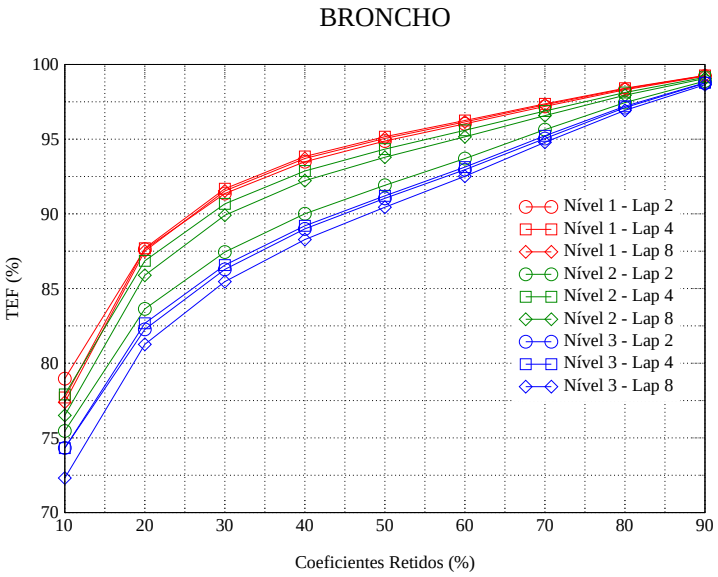


Figura 4.8: Efeito do tamanho do intervalo de sobreposição na eficiência da transformada - Volume BRONCHO.

Já o comportamento da concentração de energia com a variação do tamanho do *lap* mostrou-se ligeiramente diferente do resultado do volume 3DHEAD. Embora tenhamos verificado uma influência maior do *lap* a medida que os blocos diminuem de tamanho, verificamos que nos níveis 2 e 3 não há uma consistente diminuição da TEF

com o aumento do *lap*. Esse comportamento pode ser explicado pela grande assimetria que há no tamanho do bloco ao longo das três direções, em razão da transformada ser aplicada usando a mesma partição diádica ao longo dos três eixos e do volume possuir dimensões diferentes. Por exemplo, no nível 2, os blocos possuem tamanho 128×128×32 e, no nível 3 o tamanho é de 64×64×16. Esse resultado está mostrado no gráfico da Figura 4.8.

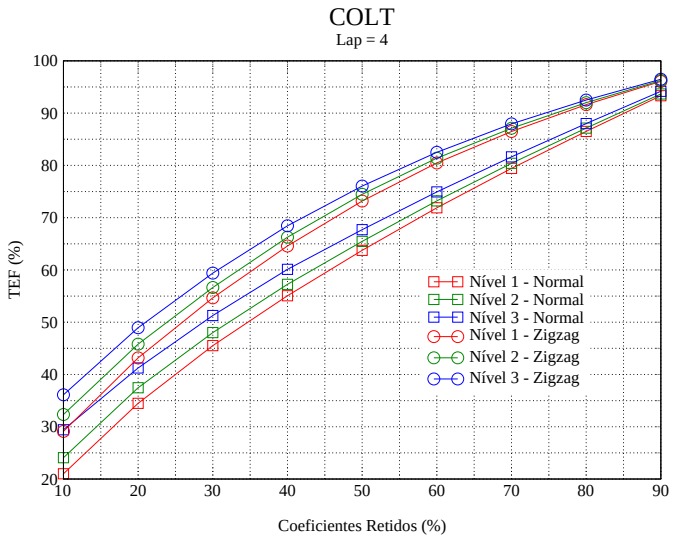


Figura 4.9: Eficiência da transformada com caminhamento normal e em zigzag - Volume COLT.

O dado de sísmica COLT apresentou os menores ganhos nos níveis de concentração de energia, quando da utilização do caminhamento em zigzag. O caminhamento em zigzag, para 10% dos coeficientes retidos, representou um ganho na TEF entre 10% e 6%. O comportamento em relação aos níveis de decomposição também se mostrou diferente, havendo um ganho na TEF quando decompomos mais o volume, conforme podemos ver no gráfico da Figura 4.9.

Verificamos também (Figura 4.10), que se mantém a maior influência do *lap* a medida que o bloco diminui de tamanho, e um aumento de até 16% quando comparamos a TEF obtida no nível 1 com a obtida no nível 4 ($lap = 4$).

O volume VH também confirma a escolha do caminhamento em zigzag, apresentando um ganho na TEF em relação ao caminhamento normal de 9%, quando representamos o volume transformado com apenas 10% dos primeiros coeficientes (Figura 4.11).

Também foi observado que a concentração de energia para esse dado diminui com o incremento do tamanho do intervalo de sobreposição, conforme demonstra a Tabela 4.2.

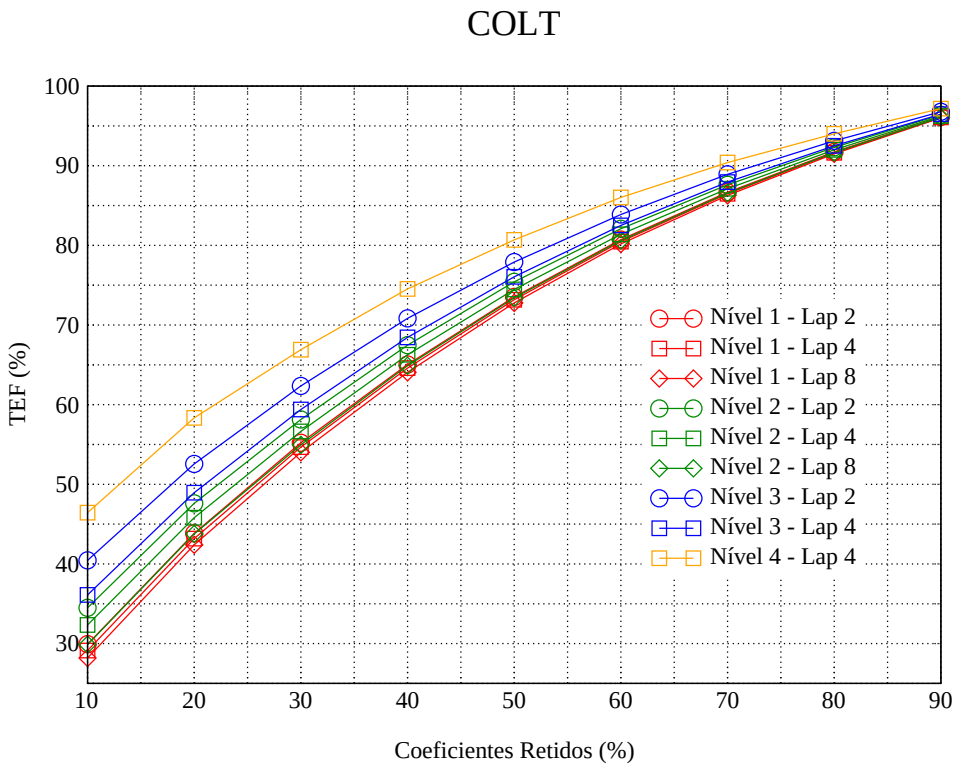


Figura 4.10: Efeito do tamanho do intervalo de sobreposição na eficiência da transformada - Volume COLT.

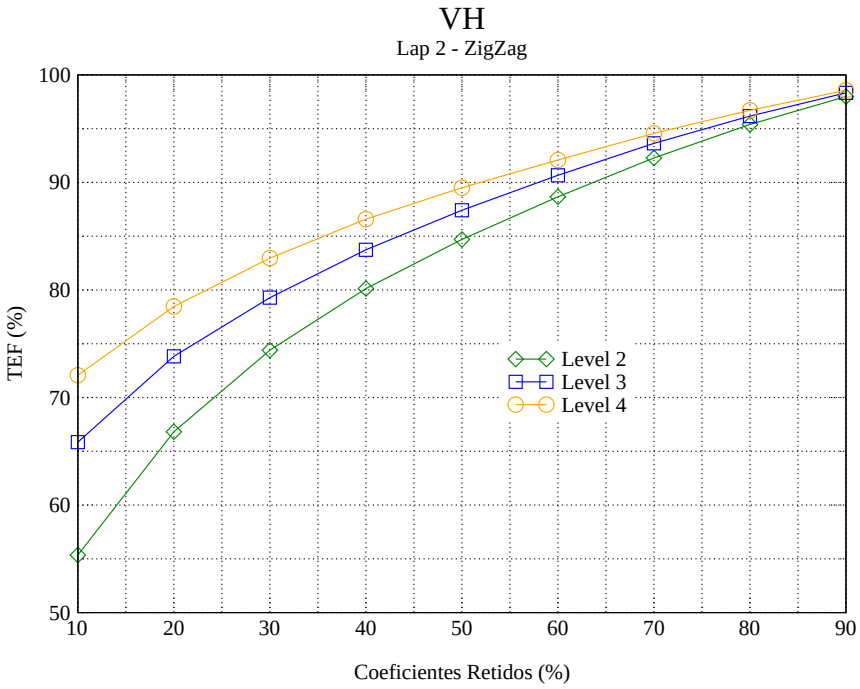


Figura 4.12: Efeito do nível de decomposição na eficiência da transformada - Volume VH.

LAP	10% Coef.	90% Coef.	Diferença 10%	Diferença 90%
2	65.8	98.3	—	—
4	62.4	98.2	3.4	0.1
8	58.2	97.4	4.2	0.3

Tabela 4.2: Eficiência da Transformada - VH - Nível de decomposição 3.

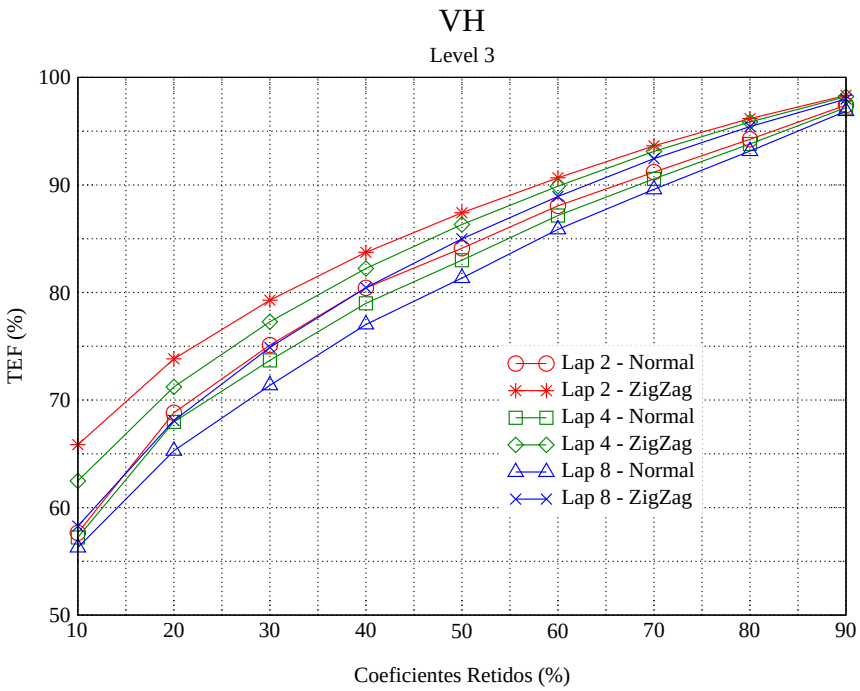


Figura 4.11: Eficiência da transformada com caminhamento normal e em zigzag - Volume VH.

A Figura 4.12 confirma a afirmativa de que a concentração de energia é um fenômeno local; no gráfico podemos verificar um aumento de até 16% na TEF entre o nível 2 (blocos com 64×64×32 *voxels*) e o nível 4 (blocos com 16×16×4 *voxels*).

Os menores ganhos na TEF com a utilização do caminhamento em zigzag foram obtidos com o conjunto de dados sísmicos, no qual a opção pelo caminhamento em zigzag representou um aumento de aproximadamente 10%. Para os volumes obtidos por ressonância magnética, o aumento na TEF ficou na faixa de 10% a 25%. Observamos também que o valor da TEF é menor nos níveis de maior decomposição, ou seja, mais discretizados. Isto significa que a TEF aumenta quando utilizamos um menor número de blocos para representar o volume. O tamanho da porção sobreposta da janela também afeta o valor da TEF. Verificamos que, dobrando o tamanho da região de sobreposição, há um descréscimo de aproximadamente 2% na TEF.

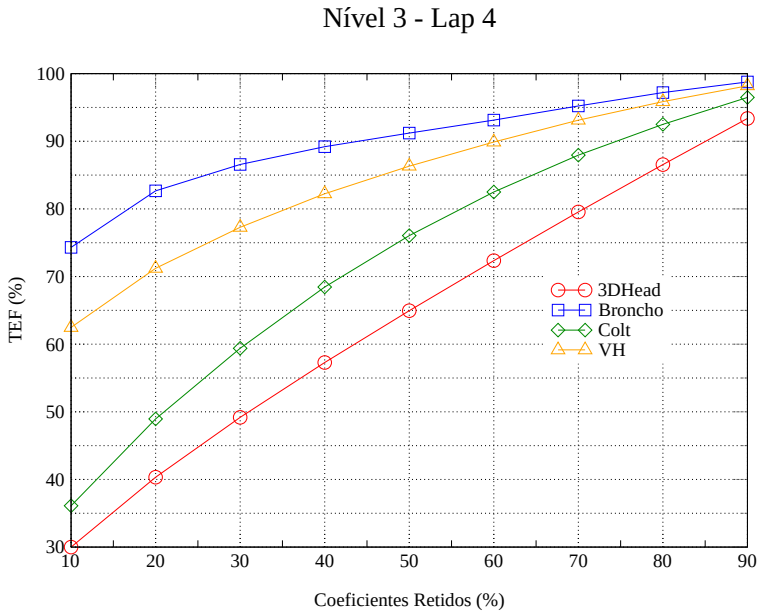


Figura 4.13: Comparação da Eficiência da Transformada - Vários Volumes.

Comparando a concentração de energia calculada para todos os conjuntos de dados, como mostrado na Figura 4.13, foi observado que a concentração de energia é maior nos conjuntos de dados que já estavam quantizados (3DHEAD, BRONCHO, COLT). Os valores dos *voxels* desses dados foram quantizados da sua representação original para valores com 8 bits por *voxel*, o que representa uma operação de filtragem passa baixa, o que atenua as altas freqüências do volume e reduz o ruído. Esse comportamento não acontece no dado VH, que foi utilizado com os valores dos *voxels* em sua representação original, 16 bits. Nos volumes adquiridos via tomografia computadorizada, foi observada a maior concentração de energia resultante desta transformada. A baixa concentração de energia apresentada pelo dado de sísmica decorre do fato desses dados serem extremamente ruidosos, com superfícies e regiões de homogeneidade pouco definidas.

4.2.3 Resultados de Compressão

Foi também realizada uma série de testes para avaliar a capacidade de compressão do método proposto, a qualidade do dado reconstruído e quantificar a distorção introduzida pela compressão. Os testes visavam avaliar a influência do nível de decomposição, tamanho da porção sobreposta dos intervalos e do número de células de quantização utilizadas para a representação dos coeficientes da transformada na razão de compressão e no erro de reconstrução.

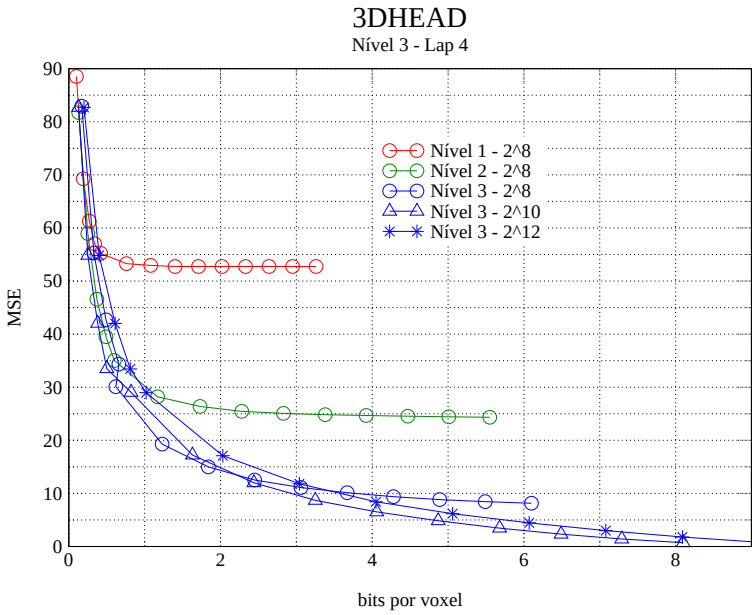


Figura 4.14: Comparação do MSE para diferentes níveis de decomposição e números de células de quantização - Volume 3DHEAD.

Para o dado 3DHEAD, verificamos que o MSE diminui à medida que aumentamos o nível de decomposição. Também foi observado que a partir de 30% dos coeficientes retidos, não há uma grande variação do MSE (resultado nâo apresentado). Isso significa que, a partir deste ponto, estamos próximos do erro devido à quantização dos coeficientes da transformada. Esta conclusão é reforçada pela diminuição do patamar de MSE, quando aumentamos o número de células de quantização. Foi também observado MSE de 1.0, utilizando 2^{10} células de quantização no nível 3 e tamanho do intervalo de sobreposição 4; diminuindo o número de células de quantização para 2^8 o MSE sobe para 8.15. Também foi verificado que a influência do número de células de quantização é maior nos menores níveis de decomposição, que correspondem a blocos maiores, como mostrado na Figura 4.14. Nessa figura, temos nos níveis 1 e 2, mais de 8 bits por *voxel*, quando usamos 70 % dos coeficientes de cada bloco quantizados com 2^{10} células de quantização, o que representa uma expansão do arquivo e não uma compressão.

Observando o efeito do tamanho do intervalo de sobreposição sobre o MSE, temos que as curvas de distorção correspondentes ao nível 2, com 2^8 células de quantização, para valores de *lap* iguais a 2, 4 e 8 são praticamente coincidentes.

O volume COLT apresenta uma maior sensibilidade ao número de células de quantização, apresentando um MSE maior que 420 no nível 2, com 2^8 células de quantização. Usando 2^{10} células, reduzimos o MSE mínimo para 84 e, utilizando 2^{12} células

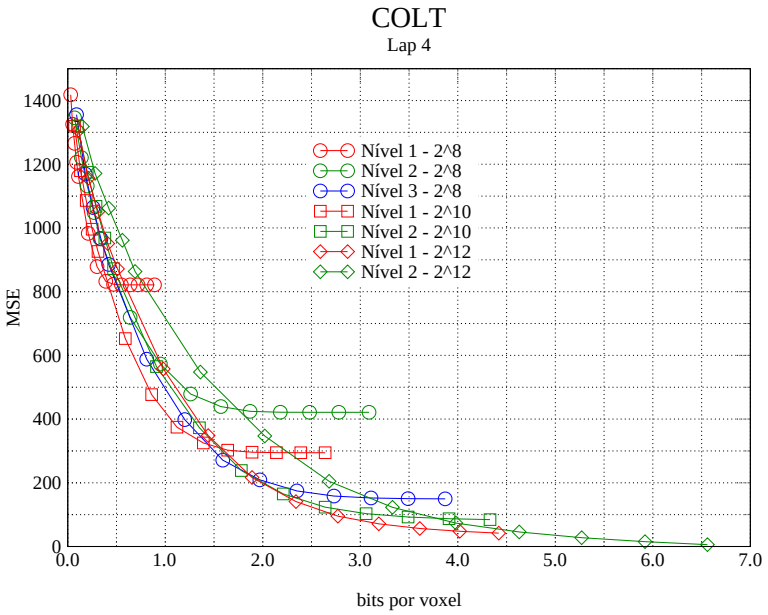


Figura 4.15: Comparação da distorção para diferentes níveis de decomposição e numero de células de quantização - Volume COLT.

esse erro atinge 5.86 (Figura 4.15). Também foi observada, para esse volume, uma influência significativa do tamanho do *lap* no MSE, de modo a reduzir a taxa de compressão do volume quando aumentamos o *lap* mantendo o mesmo nível de distorção (resultado não apresentado).

Analizando o gráfico da Figura 4.16, verificamos a influência do número de células de quantização nos resultados de compressão. Novamente temos que o uso de 2^8 células não parece apropriado para os níveis de menor decomposição do volume. Por outro lado, a redução obtida no erro em razão do aumento do número de classes de quantização, decresce significativamente a medida que aumentamos o nível de decomposição. Esse resultado é esperado, pois usamos uma quantização baseada nos blocos do volume transformado, nos quais usamos 2^c células para representar os coeficientes. Assim, nível de decomposição maior significa blocos menores ou menor número de coeficientes para serem quantizados, o que faz com que a quantização com menor número de células introduza um erro menor. No nível de decomposição 1, a utilização de 2 bits adicionais para quantizar os coeficientes resulta num decréscimo de 80 unidades no MSE, mas no nível 3, esse bits adicionais decrescem o MSE de menos de 20 unidades.

Aplicada a compressão ao dado VH, foram obtidas respostas similares aos anteriores. A razão de distorção mostra uma tendência de atingir o valor mínimo de MSE a partir da utilização de 30% dos coeficientes da transformada em cada bloco. Novamente, verificamos a necessidade de mais células de quantização nos níveis menores

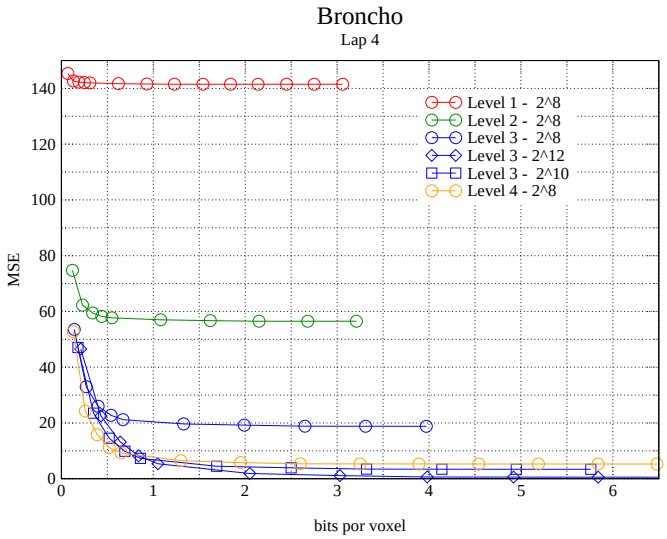


Figura 4.16: Comparação da distorção para diferentes níveis de decomposição e número de células de quantização - Volume BRONCHO.

de decomposição, ou então, de um método de quantização mais apropriado que a quantização uniforme. Analizando o PSNR, verificamos que o erro de reconstrução diminui, mantendo uma taxa de compressão fixa, à medida que aumentamos o nível de decomposição. Esses resultados confirmam a suposição de que a correlação é um fenômeno local, e são obtidos também em virtude da diminuição da razão entre o tamanho nominal do bloco de volume transformado e o tamanho do *lap*. As curvas PSNR×bpv apresentam um aumento do PSNR máximo à medida que aumentamos o nível de decompsição. Por exemplo, no nível 2, o PSNR máximo é 41 dB, sendo de 46 dB no nível 3, e de 52 dB no nível 4. Esses valores correspondem a 2.1, 1.9 e 2.07 bpv, respectivamente. A Figura 4.17 apresenta essas curvas.

Para um percentual fixo de coeficientes mantidos, verificamos que o nível de decomposição 2 apresenta uma taxa de compressão maior, o que significa que, nesse nível, encontramos uma maior concentração de energia para esse volume.

Utilizando 2^{12} células de quantização, observamos que, para altas taxas de compressão ($bpv \leq 1.5$), os resultados eram melhores que com a utilização de 2^{10} células. Este resultado se inverte na zona correspondente às taxas menores de compressão ($bpv > 1.5$). A utilização de 2^8 células de quantização, mais uma vez se mostrou inapropriada, apresentando MSE maior que 1000 e PSNR menor que 39.

Foi verificado que os melhores resultados da aplicação da transformada advém dos dados de CT, enquanto os dados de sísmica apresentaram a pior taxa de compressão. Este resultado é apresentado na Figura 4.18.

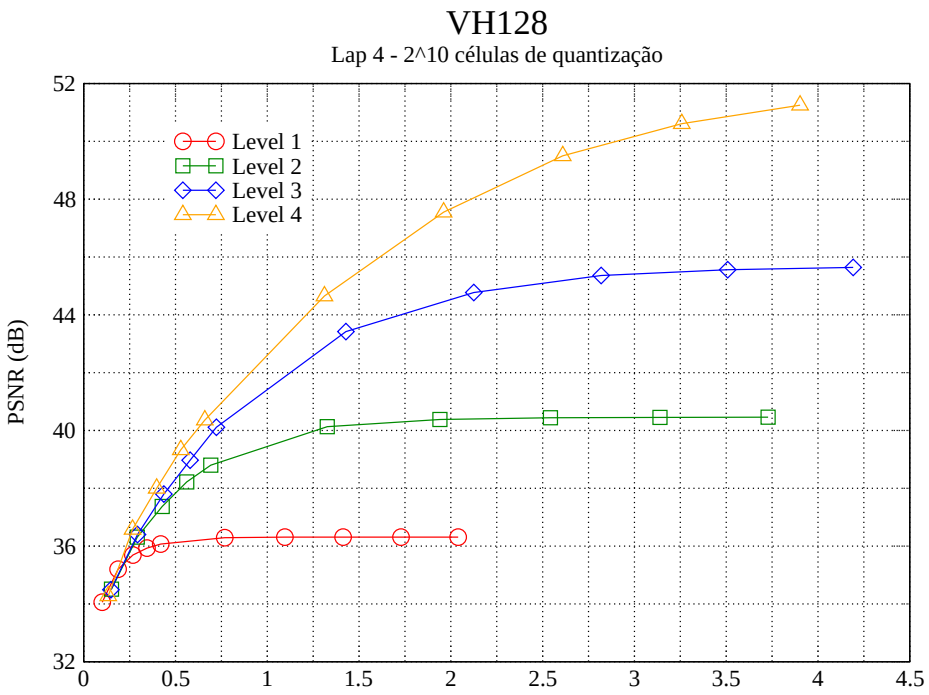


Figura 4.17: Comparação de PSNR em diferentes níveis de decomposição - Volume VH.

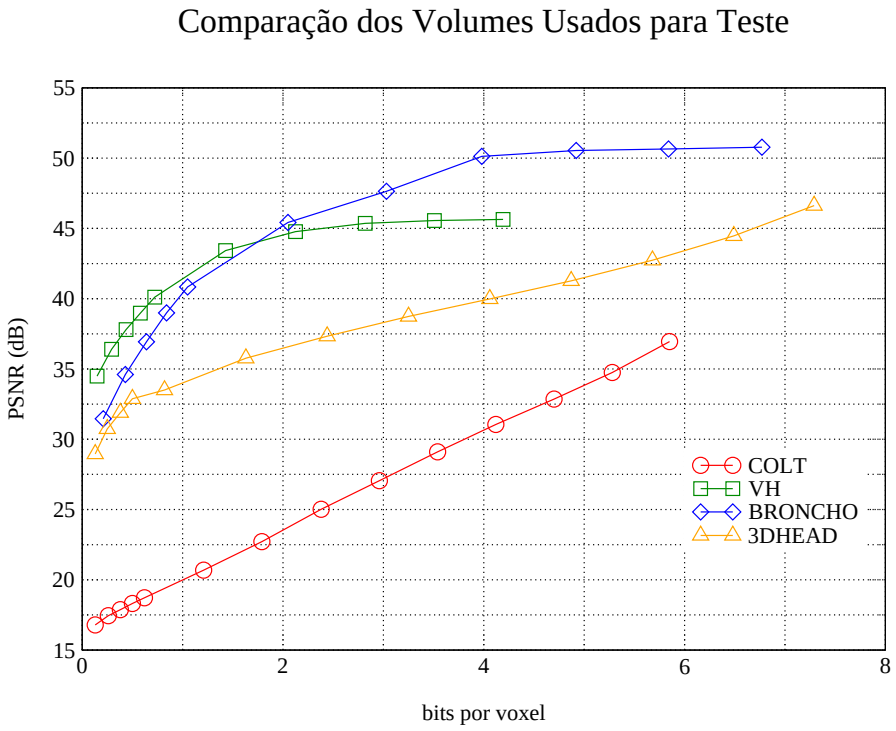


Figura 4.18: Comparação do PSNR dos dados utilizados para testes (bits por voxel).

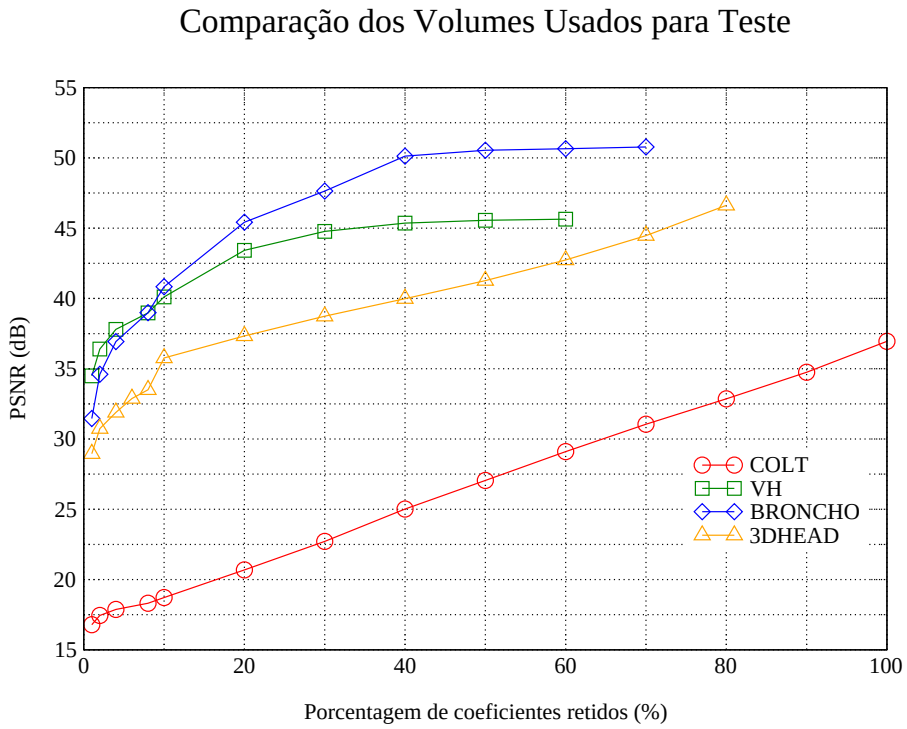


Figura 4.19: Comparação do PSNR dos dados utilizados para testes (Porcentagem de coeficientes retidos).

4.2.4 Resultados Utilizando *Best Basis*.

Os resultados obtidos com o algoritmo de *best basis* indicaram que somente obtemos resultados efetivos quando utilizamos toda a árvore de cossenos locais para realizar a escolha das melhores bases. Assim, verificamos que é melhor utilizar o algoritmo de *best basis* com um tamanho pequeno para o intervalo de sobreposição, de modo a permitir que os níveis de maior decomposição do volume sejam incluídos na busca pelas melhores bases.

No entanto, foi observado que, exceto na zona de altíssimas taxas de compressão, a solução fornecida pelo algoritmo de *best basis* representa um ganho de compressão para um mesmo nível de distorção, conforme observado nas Figuras 4.20 e 4.21. As curvas correspondentes ao volume COLT indicam que os resultados obtidos pelo algoritmo de *best basis* apresentam um aumento de 2 dB com relação ao melhor resultado obtido com nível de decomposição fixo, a partir de $bpv > 2.5$. Para o volume 3DHEAD verificamos que este ponto de mudança de comportamento se dá para $bpv > 1.3$. Isto indica que mesmo com a necessidade adicional de codificar a descrição dos blocos, temos um acréscimo na capacidade de compressão em razão da adaptação da decomposição às características do volume.

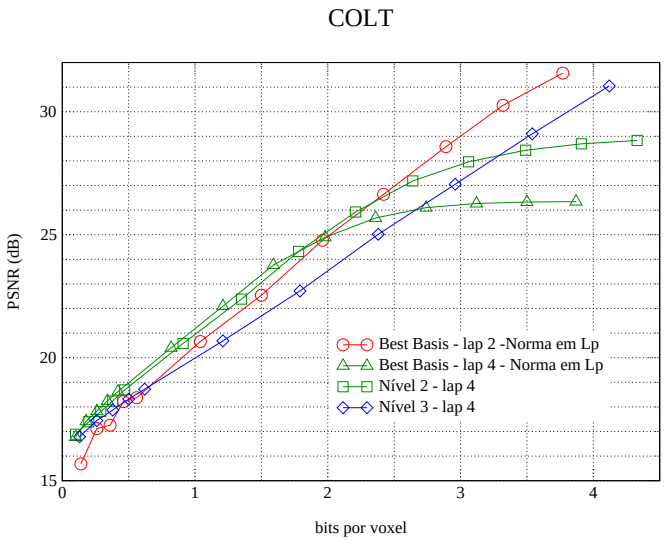


Figura 4.20: Comparação da abordagem *best basis* e nível fixo - Volume COLT .

Adicionalmente, verificamos que esta abordagem evita que o usuário da compressão necessite experimentar com diferentes situações para encontrar a melhor combinação nível de decomposição/*lap*. Também foi verificado que o funcional utilizado para calcular o custo de cada base tem grande influência sobre os resultados do algoritmo de *best basis*. Por exemplo, para o dado COLT a Norma em L^p foi a que apresentou os melhores resultados para efeito de compressão.

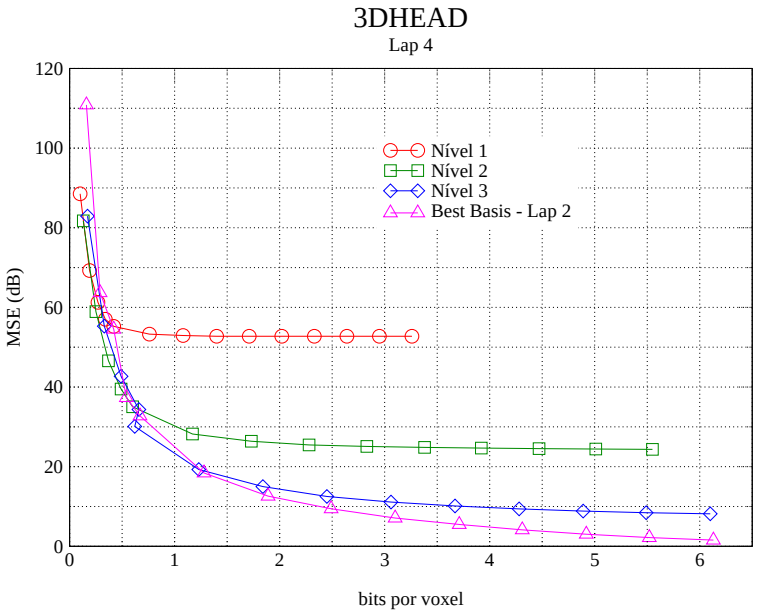


Figura 4.21: Comparação da abordagem *best basis* e nível fixo - Volume 3DHEAD .

4.2.5 Comparação com Outros Métodos de Compressão

Os resultados do volume VH foram comparados com os obtidos em outros trabalhos. Para isso foram utilizadas três versões diferentes do volume VH original. O volume VH128 já descrito, uma versão denominada VHLOW, que foi formada pelas primeiras 256 fatias do volume original escaladas [2] para 256×256 *voxels*, sendo, portanto, de tamanho $256 \times 256 \times 256$, e VHTOTAL, obtido por uma escala ao longo das três direções para o tamanho $256 \times 256 \times 256$.

O objetivo desses testes era avaliar a influência da obtenção de blocos de volume mais regulares. Foi verificado, como mostrado no gráfico da Figura 4.22, que os dados mais regulares apresentavam uma performance superior na região de alta taxa de compressão ($bpv < 1$), apresentando uma tendência a atingir um maior PSNR para baixas taxas de compressão. No entanto, podemos verificar que a diferença na taxa de compressão para um mesmo nível de PSNR é bastante pequena, levando-nos a concluir que as escalas realizadas no dado não alteraram de maneira significativa o seu comportamento quanto à compressão.

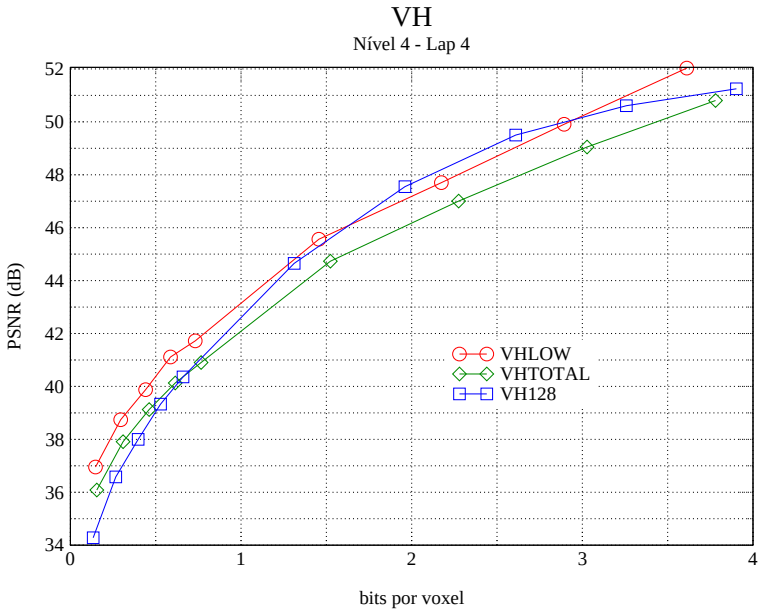


Figura 4.22: Comparação de diferentes cortes do volume VH original.

Os resultados obtidos com o volume VH recortado foram comparados com os resultados obtidos pelos métodos propostos em (Kim & Shin, 1999), (Ihm & Park, 1998) e (Rodler, 1999). Incialmente comparamos a compressão conforme proposta

[2] Realizada através da média dos valores vizinhos

na seção anterior, e foi obtida a curva mostrada no gráfico da Figura 4.23, curva correspondente a Bloco 8x8x8, a qual corresponde a uma decomposição no nivel 5 do volume VHLOW, tendo sido utilizado o tamanho do intervalo de sobreposição como 2. Podemos verificar que este resultado apresenta comportamento similar ao apresentado em (Ihm & Park, 1998), sendo superior ao método proposto em (Kim & Shin, 1999). Ao compararmos com os resultados apresentados em (Rodler, 1999), verificamos uma diferença de 0.25 a 0.6 bits por *voxel* na taxa de compressão para o mesmo nível de erro de reconstrução (PSNR). No entanto, esse ganho de compressão existente no método proposto em (Rodler, 1999) é devido à utilização de algoritmos preditivos entre fatias do volume, o que prejudica sobremaneira a tarefa de desenvolvimento de um algoritmo de visualização baseado na representação comprimida do dado.

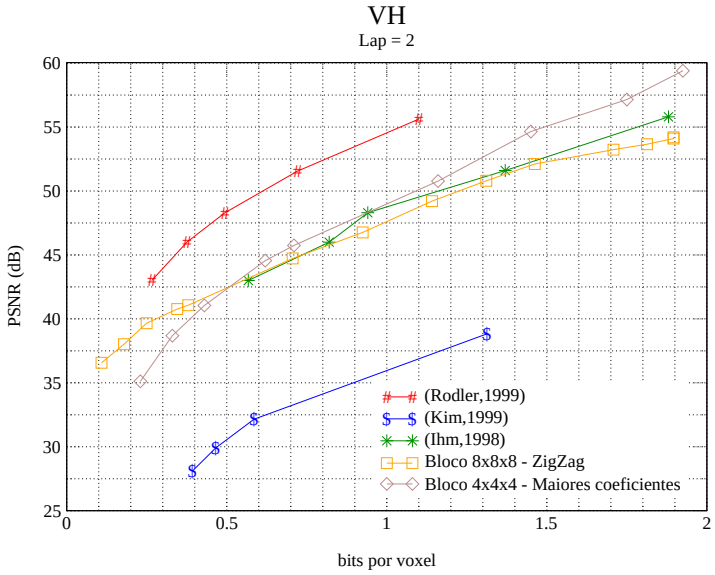


Figura 4.23: Comparação dos resultados de compressão com outros métodos.

Uma outra comparação foi realizada com o método proposto em (Ihm & Park, 1998). Foi calculado o erro de reconstrução obtido com a manutenção de $\tau\%$ dos maiores coeficientes da transformada em cada bloco, após a quantização. Os $\tau\%$ coeficientes retidos foram escolhidos de duas maneiras: os primeiros coeficientes obtidos segundo o caminhamento em zigzag e os maiores coeficientes. Verificamos que a escolha dos maiores coeficientes de cada bloco, ao invés dos $\tau\%$ primeiros, diminui o erro de reconstrução. Este resultado está apresentado na Tabela 4.3, que mostra os valores do PSNR, medido em decibéis, apresentados no artigo citado, e os valores obtidos com o método proposto neste trabalho.

Com base nesse resultado, foi preparado um experimento para verificar o com-

	τ% : Coeficientes Retidos (%)			
	3%	5%	7%	10%
(Ihm & Park, 1998)	43.0	46.0	48.3	51.6
VH128 (ZigZag)	34.5	36.6	38.9	40.1
VH128 (Maiores)	41.4	43.6	45.3	46.4

Tabela 4.3: Erro de Reconstrução (PSNR(dB)) - VH.

portamento de compressão possível com a abordagem de escolha dos maiores coeficientes de cada bloco. Assim, foi desenvolvido um esquema de compressão que se baseia na divisão do volume em blocos de $4 \times 4 \times 4$ *voxels*. Em cada *voxel* é aplicada uma transformada tridimensional do cosseno local com o tamanho do intervalo de sobreposição definido como 2. Em seguida, é aplicada a quantização dos coeficientes da transformada de cada bloco usando as equações 4.1 e 4.2. Então, são selecionados os τ% maiores coeficientes, os quais são codificados, juntamente com os valores necessários à quantização inversa: o passo de quantização e o valor do menor coeficiente do bloco. Adicionalmente, é necessário codificar a posição dos coeficientes mantidos, o que é feito através do mapa de posição (*significance map*). Esta codificação foi feita em um vetor de posição ($coef_{pos}$) com 64 bits (4x4x4 posições no bloco), cada um indicando se o coeficiente correspondente foi mantido ou não. O vetor $coef_{pos}$ é armazenado em até 8 bytes. Para evitar que os 8 bytes do vetor $coef_{pos}$ sejam escritos para todos os blocos, utilizamos um byte *pos* que indica quais bytes do vetor $coef_{pos}$ possuem pelo menos um bit igual a 1, e necessitam serem escritos. Após a codificação da posição, os coeficientes escolhidos, juntamente com os bytes de codificação de suas posições são codificados, usando um algoritmo de compressão sem perda. No experimento realizado esses coeficientes são escritos em um arquivo sobre o qual é aplicado o compactador gzip.

O resultado correspondente a este experimento está representado pela curva "Bloco 4x4x4-Maiores" da Figura 4.23. Nessa curva, podemos verificar o que foi mostrado por Shapiro (Shapiro, 1996), que os bits necessários para codificar o mapa de posição podem consumir uma grande parte da economia de bits, ficando essa situação mais crítica nos níveis de maior compressão. A partir de 0.6 bpv, temos uma considerável queda da curva razão distorção em razão do aumento da relação entre o número de bits codificando coeficientes da transformada, que realmente diminui, e o número de bits codificando a posição dos coeficientes retidos, o qual diminui em uma razão muito menor.

Observa-se, portanto, que a compressão de volumes com a transformada do

cosseno local, utilizada com um cuidadoso algoritmo de codificação, pode apresentar resultados de compressão ainda melhores, mostrando-se uma promissora abordagem para compressão de volumes. Foi observado, nos experimentos realizados, que o método se adequa facilmente aos dados de tomografia, sendo menos eficaz nos dados sísmicos.

Capítulo 5

Conclusões

Esta tese descreve um método para a compressão de dados volumétricos baseado na transformada do cosseno local. O método de compressão proposto representa um algoritmo promissor para a compressão de dados volumétricos, o qual permite a descompressão bloco a bloco do volume, sendo potencialmente adequado ao desenvolvimento futuro de um algoritmo de visualização que possa permitir a um grande número de usuários a visualização de volumes com grandes dimensões.

Apresentamos, nesta tese, um estudo sobre o comportamento dessa transformada para a compressão, de onde pudemos concluir que as melhores características de compressão são obtidas quando trabalhamos com blocos de volume de pequenas dimensões, tratando a correlação como um fenômeno o mais local possível. Por outro lado, foi verificado que, em algumas regiões do volume, as características de homogeneidade, ou, no caso de dados médicos, a existência de grandes vazios, podem ser exploradas de modo a aumentar a capacidade de compressão do dado. Assim, verificamos que o algoritmo de *best basis* apresenta características de compressão mais vantajosas que o método do nível de decomposição fixo, embora sua utilização em um ambiente de visualização se torne mais complexa, em virtude da generalização das adjacências entre blocos, que aumenta a complexidade do algoritmo.

Foi observado que a quantização é uma etapa central no processo de compressão do volume. Neste trabalho foi utilizada uma abordagem de quantização uniforme que se tornou efetiva, quando utilizada com um certo grupo de parâmetros. Para blocos de tamanho superior a 32x32x32, é necessária a utilização de um mínimo de 2^{10} células, de modo evitar que o processo de quantização introduza um considerável erro na reconstrução. Verificamos que, com esse número de células, ainda mantemos características razoáveis de compressão. Quando se faz necessário um aumento na taxa de compressão, com a manutenção de pequenos erros de reconstrução, podemos utilizar a abordagem de blocos pequenos, tipicamente 4^3 ou 8^3 *voxels*, associada a um número menor de células de quantização.

Quando comparado a outros trabalhos, o método proposto necessitou de um cuidado maior para atingir os níveis de compressão desejados. No entanto, com pouca elaboração, foi possível adequar um processo de quantização/codificação apropriado à transformada, que desse respostas próximas dos métodos existentes. Assim, veri-

ficamos que podemos desenvolver um eficiente método de compressão através do desenvolvimento e melhoria de um método baseado na escolha dos maiores coeficientes de cada bloco, desde que consigamos uma estratégia eficiente para codificar o mapa de posições desses coeficientes. O método aqui proposto apresentou resultados satisfatórios, com a desvantagem de consumir uma porção razoável de bits para codificar os parâmetros necessários à quantização inversa (passo de quantização e valor do coeficiente de freqüência mínimo).

Como vantagens adicionais desse método de compressão podemos citar a facilidade de implementação de um algoritmo de visualização baseado na ordem dos objetos que se acople ao algoritmo de reconstrução local. Isto pde ser verificado, posto que ao realizar a descompressão local do volume, o que pode ser feito segundo qualquer direção de caminhamento, temos um subvolume que pode ser submetido a um processo de visualização como um volume. Também podemos concluir que o método é apresenta características adequadas à sua paralelização, que pode ser impelementada de maneira simples, descomprimindo e visualizando o volume bloco a bloco.

Como principais contribuições desta tese podemos citar:

- A aplicação da transformada do cosseno local a dados volumétricos
- Desenvolvimento de um algoritmo de reconstrução bloco a bloco do dado volumétrico comprimido, facilmente incorporável a um método de visualização existente.
- Estudo do comportamento da transformada do cosseno local quando aplicada a dados volumétricos.

5.1 Sugestões para Trabalhos Futuros.

Neste trabalho, procuramos investigar o comportamento da transformada local, quando aplicada a dados volumétricos. Como extensão direta desse trabalho, podemos citar o desenvolvimento de um algoritmo de visualização de dados volumétricos apropriado à estrutura do dado comprimido, que permita a visualização apenas com a descompressão parcial do volume. Acredita-se que os métodos da classe da ordem dos objetos sejam mais adequados, em virtude de necessitarmos realizar a descompressão dos blocos do volume em uma ordem apropriada, para, em seqüência ir atribuindo as compressões das porções sobrepostas entre vizinhos. Duas escolhas naturais para essa extensão são os algoritmos *Splatting* (Westover, 1990) e *Shear Warping* (Lacroute & Levoy, 1995).

Seguindo o desenvolvimento do método de compressão, algumas investigações adicionais se fazem necessárias, como um estudo do efeito das perdas sobre a visuali-

zação dos dados. Também se faz necessária uma investigação dos métodos de codificação dos coeficientes da transformada, para determinar qual o mais apropriado, assim como o estudo de diferentes estratégias de quantização para determinar a que resulta em minimização dos erros de reconstrução. No sentido de aumentar a capacidade de compressão do método proposto, poderia ser estudada a aplicação de uma variante da transformada do cosseno local, substituindo a DCT-IV por uma outra transformada que apresentasse a propriedade de resolução de constante (e.g. DCT-II), representando com mais eficiência áreas homogêneas do volume.

Também se faz necessária uma melhor calibragem do algoritmo de *best basis*, de modo a definir funções de custo que correspondam melhor às características de compressão dos blocos, além do desenvolvimento de um algoritmo de reconstrução local baseado na representação em *best basis*.

Uma outra variação desejável do método proposto é a possibilidade de utilizar um algoritmo de *best basis* baseado em uma decomposição do volume mais genérica que a decomposição diádica, e que permita a utilização de diferentes tamanhos para o intervalo de sobreposição ao longo das fronteiras entre os blocos do sinal.

A aplicação do método de compressão proposto a várias classes de volume se faz necessária, aprofundando o estudo do comportamento da transformada com cada um deles. No caso de dados provenientes de levantamentos sísmicos, seria desejável um estudo com comparação do efeito da utilização da transformada 3D em relação a compressão 1D e 2D, comumente utilizada nessa área. Além disso, se faz necessário estimar o efeito das perdas provenientes da compressão em pós-processamento desses dados sísmicos, tais como migração.

Referências Bibliográficas

Abramson, N. 1963. *Information Theory and Coding.* NewYork: McGraw Hill.

Ackerman, M. A. 1996. Visible Woman: High Resolution Companion to the Visible Man. *Interactive Healthcare Newsletter*, **12**(7/8), 5–6.

Aharony, G.; Averbuch, A.; Coifman, R. R. & Israeli, M. 1993. Local Cosine Transform - A method for the Reduction of the Blocking Effect in JPEG. *Pages 205–217 of:* Laine, A. F. (ed), *Mathematical Imaging: Wavelet Applications in Signal and Image Processing*, vol. 2034.

Aharony, G.; Averbuch, A.; Coifman, R. R. & Israeli, M. 1994. Local Cosine Transform - A method for the Reduction of the Blocking Effect in JPEG. *Journal of Mathematics and Image Vision*, **3**, 7–38.

Ahmed, N. & Rao, K. R. 1974. Discrete Cosine Transform. *IEEE Transaction on Computers*, **C-23**(January), 90–93.

Auscher, P.; Weiss, G. & Wickerhauser, M. V. 1992. *Local Sine and Cosine Bases of Coifman and Meyer and the Construction of Smooth Wavelets.* Academic Press. In: Wavelets - A Tutorial in Theory and Applications, Edited by Chui, C. K. Pages 237–256.

Barnsley, M. F. & Demko, S. G. 1985. Iterated function schemes and the global construction of fractals. *Pages 243–275 of: Proceedings of Royal Society A*, vol. 399.

Barthold, L.; Crane, R. & Naqvi, S. 1997. *Introduction to Volume Rendering.* Addison-Wesley.

Brocklehurst, A. 1995. *Data Compression.* Tech. rept. British Petroleum Exploration.

Chan, K. K.; Lau, C. C.; Chuang, K. S. & Morioka, C. A. 1991. Visualization and Volumetric Compression. *Image Capture, Formatting and Display*, **SPIE 1444**, 250–255.

Chen, W.-H. & Pratt, W. K. 1984. Scene Adaptive Coder. *IEEE Transactions on Communications*, March, 225–232.

Chen, Y. & Pearlman, W. A. 1996 (March). Three-dimensional Sub-band Coding of Video Using the Zero-tree Method. *Pages 1302–1310 of: Proceedings of SPIE-Visual Communications and Image Processing '96.*

Chiueh, T.-C.; Yang, C.-K.; He, T.; Pfister, H. & Kaufman, A. 1997 (october). Integrated Volume Compression and Visualization. *In: IEEE Visualization'97.*

Coifman, R. R. & Wickerhauser, M. V. 1992. Entropy-based Algorithms for Best Basis Selection. *IEEE Transactions on Information Theory*, **38**(2), 713–718.

Coifman, R. R.; Meyer, Y. & Wickerhauser, M. V. 1991. *Wavelet Analysis and Signal Processing.* Boston: Jones and Bartlett. In: Wavelet and Their Applications, edited by B. Ruskai et al. Pages 153–178.

Danskin, J. & Hanrahan, P. 1992. Fast Algorithms for Volume Rendering. *Pages 91–98 of: ACM Workshop on Volume Visualization 1992.*

Daubechies, I. & Lagarias, J. C. 1991. Two Scale Difference Equation: Existence and Global Regularity of Solutions. *SIAM Journal on Mathematics Analysis*, **22**(5), 1388–1410.

Duhamel, P.; Mahieux, Y. & Petit, J. 1991. A Fast Algorithm for the Implementation of Filter Banks Based on Time Domain Aliasing Cancellation. *Pages 2209–2212 of: Proc. of IEEE Intrenational Conference on Accustic, Speech and Signal Processing.*

Dunne, S.; Nappel, S. & Rutt, B. 1990. Fast Reprojection of Volume Data. *Pages 11–18 of: Proc. of the 1st Conference on Visualization in Biomedical Computing.*

Fowler, J. E. & Yagel, R. 1994. Lossless Compression of Volume Data. *Pages 43–50 of: ACM SIGGRAPH Symposium on Volume Visualization.*

Gomes, J.; Costa, B.; Darsa, L. & Velho, L. 1996. Graphical Objects. *The Visual Computer*, **12**(6), 269–282.

Gross, M. H.; Lippert, L. & Staadt, O. G. 1998. *Compression Methods for Visualization.* Tech. rept. 293. ETH- Swiss Federal Institute of Technology Zurich.

Grosso, R.; Ertl, Th. & Aschoff, J. 1996 (February). Efficient Structures for Volume Rendering of Compressed Data. *In: WSCG'96 - The Fourth International Confference in Central Europe on Computer Graphics and Visualization.*

Guazzo, M. 1980. A General Minimum-Redundancy Source-Coding Algorithm. *IEEE Transactions on Information Theory*, **26**(January), 15–25.

Haley, M. B. & Blake, E. H. 1996. Incremental Volume Rendering Using Hierarchical Compression. *Computer Graphics Forum*, **15**(3), 44–55.

Howard, P. G. & Vitter, J. S. 1993 (March). Fast and efficient losless image compression. *Pages 351–360 of: 1993 Data Compression Conference.*

Huffman, D. A. 1952. A Method for the Construction of Minimum Redundancy Codes. *Proceedings IRE*, **40**(10), 1098–1101.

Ihm, I. & Park, S. 1998. Wavelet-Based 3D Compression Scheme for Very Large Volume Data. *Pages 107–116 of: Graphics Interface'98.*

Jacquin, A. E. 1992. Image coding based on fractal theory of iterated contractive image transformation. *IEEE Transactions on Image Processing*, **1**(1), 18–30.

Jawerth, B.; Liu, Y. & W., Sweldens. 1994. Signal Compression with Smooth Local Trigonometric Bases. *Optical Engineering*, **33**, 2125–2135.

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

Kim, T.-Y. & Shin, Y. G. 1999 (October). An Efficient Wavelet-Based Compression Method for Volume Rendering. *In: Pacific Graphics'99.*

Kim, Y. S. & Pearlman, W. A. 1999. Losless Volumetric Image Compression. *Pages 305–312 of: Applications of Digital Image Processing XXII, Proceedings of SPIE*, vol. 3808.

Knittel, G. 1995 (October). High-speed volume rendering using Redundant Block Compression. *In: Proceedings of the IEEE Visualization '95 Conference.*

Lacroute, P. & Levoy, M. 1995. Fast Volume Rendering using a Shear-Warp Factorization of the Viewing Transformation. *Computer Graphics*, **28**(3), 451–458.

Laeng, E. 1990. Une Base Orthonormale de $L^2(\mathbb{R})$, dont les Éléments Sont Bien Localisés dans l'Espace de Phase et Leurs Supports Adapté à Toute Partition Symétrique de l'Espace des Fréquences. *Comptes Rendus de l'Académie des Science de Paris*, 677–680.

Levoy, M. 1990. Efficient Ray-Tracing of Volume Visualization Algorithms. *ACM Transactions on Graphics*, **9**(3), 245–261.

Malvar, H. S. 1988 (June). The LOT: A Link Between Block Transforms Coding and Multirate Filter Banks. *Pages 835–838 of: Proceedings of the IEEE International Symposium on Circuits and Systems.*

Malvar, H. S. 1990. The LOT: Transform Coding Without Blocking Effects. *IEEE Transactions on Acoustic, Speech, and Signal Processing*, **38**, 969–978.

Malvar, H. S. 1992. *Signal Processing with Lapped Transform.* Norwood, MA.: Artech House.

Malvar, H. S. & Staelin, D. H. 1989. Lapped Transforms for Efficient Transform/Subband Coding. *IEEE Transactions on Acoustic, Speech, and Signal Processing*, **37**(4), 553–559.

Malzbender, T. & Kitson, F. 1991. A Fourier Technique for Volume Rendering. *Pages 305–316 of: Focus on Scientific Visualization.*

Menegaz, G.; Grewe, L.; Lozano, A. & Thiran, J.-Ph. 1999 (September). Application oriented wavelet-based coding of volumetric medical data. *Pages 84–85 of: Proceedings of the World Congress on the Internet in Medicine (MEDNET'99).*

Meyer, F. G. 1999. Fast Compression of Seismic Data with Local Trigonometric Bases. *Pages 648–658 of:* Unser, M. A.; Aldroubi, A. & Laine, A. F. (eds), *Wavelet Applications in Signal and Image Compression VII*, vol. 3813.

Moffat, A. 1999. An improved data structure for cummulative probability tables. *Software Practice and Experience*, **29**(7), 647–659.

Moffat, A.; Neal, R. & Witten, I.H. 1998. Arithmetic Coding Revisted. *ACM Transactions on Information Systems*, **16**(3), 256–294.

Muraki, S. 1992 (October). Aproximation and Rendering of Volume Data Using Wavelet Data Fields. *Pages 21–28 of: IEEE Visualization'92 Conference Proceedings.*

Muraki, S. 1993. Volume Data and Wavelet Transform. *IEEE Computer Graphics and Applications*, **13**(4), 50–56.

Nelson, Mark. 1991. *THe Data Compression Book.* Prentice Hall.

Ning, P. & Hesselink, L. 1993. Fast Volume Rendering of Compressed Data. *Pages 11–18 of: Visualization 1993.*

NLM. Web Page for the National Library of Medicine's Visible Human Project. http: www.nlm.nih.gov/research/visible/visible_human.html.

Parkinson, C. N. 1957. *Parks's Law and Other Studies in Administration.* New York: Ballantine Books.

Pasco, R. 1976. *Source Coding Algorithms for Fast Data Compression.* Ph.D. thesis, Stanford University.

Princen, J.P. & Bradley, A. B. 1986. Analysis/Synthesis Filter Bank Design Based on Time Domain Aliasing Cancelation. *IEEE Transactions on Acoustics, Speech and Signal processing*, **34**(5), 1153–1161.

Rissanem, J. J. & Langdon, G. G. 1979. Arithmetic Coding. *"IBM"Journal of Research and Development*, March, 149–162.

Rissanem, J. J. & Langdon, G. G. 1981. Universal Modeling and Coding. *IEEE Transactions on Information Theory*, **27**(1), 12–22.

Rodler, F. R. 1999. *Wavelet Based 3D Compression for Very Large Volume Data Supporting Fast Random Acces.* Tech. rept. BRICS RS-99-34. Department of Computer Science, University of Aarhus, Aarhus C, Denmark. available at http://www.brics.dk.

Rubin, F. 1979. Arithmetic Stream Coding Using Fixed Precision Registers. *IEEE Transactions on Information Theory*, **25**(November), 672–675.

Said, A. & Pearlman, W. A. 1996. A New, Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees. *IEEE Transaction on Circuits and Systems for Vídeo Technology*, **6**, 243–250.

Sayood, K. 1996. *Introduction to Data Compression.* San Francisco, CA: Morgan Kaufmann Publishers.

Shannon, C. E. 1948. A Mathematical Theory of Communication. *Bell System Technical Journal*, **27**(3), 379–423.

Shapiro, J. M. 1996. Embedded Image Coding Using Zerotrees of Wavelet Coefficients. *IEEE Transactions on Signal Processing*, **41**(12), 3445–3462.

Taubman, D. & Zakhor, A. 1994. Multirate 3D Subband Coding of Video. *IEEE TRansactions on Image Processing*, **3**(5), 572–588.

Thoma, G. R. & Long, L. R. 1997. Compressing and Transmitting Visible Human Images. *IEEE Multimedia*, **4**(2), 36–45.

Wang, Y.; Wu, R.-S. & Jin, S. 1998 (September). Adapted Locla Cosine Transform Application to Seismic Data Compression. *Pages 47–50 of: The First Symposium on Applied Geophysics.*

Weinberger, M. J.; Seroussi, G. & Sapiro, G. 1996 (March). LOCO-I: A low complexity context based losless image compression algorithm. *Pages 140–149 of: 1996 Data Compression Conference.*

Welch, T. A. 1984. A Technique for High Performance Data Compression. *IEEE Computer*, **17**(6), 9–19.

Westermann, R. 1994 (October). A Multiresolution Framework for Volume Rendering. *Pages 51–58 of: 1994 Symposium on Volume Visualization.*

Westover, L. 1990. Footprint Evaluation for Volume Rendering. *Computer Graphics (Proc. SIGGRAPH)*, **24**(4), 367–376.

Whilhelms, J. & Gelder, A. V. 1994. Multidimensional Trees for Controlled Volume Rendering and Compression. *Pages 27–34 of: ACM Siggraph Symposium on Volume Visualization.*

Wickerhauser, M. V. 1992. High-resolution Still Picture Compression. *Digital Signal Processing: A Review Journal*, **2**(4), 204–226.

Wickerhauser, M. V. 1993 (October). Comparison ofPicture Compression Methods: Wavelet, Wavelet Packet and Local Cosine Transform Coding. *Pages 14–20 of:* Chui, C. K.; *et al.* (eds), *Wavelets: Theory, Algorithms and Applications.*

Wickerhauser, M. V.; Farge, M.; Goirand, E. & Wesfreid, E. 1994 (October). Efficiency Comparison of Wavelet Packet and Adapted Local Cosine Bases for Compression of a Two Dimensional Turbulent Flow. *Pages 509–531 of:* C. K. Chui, L. Montefusco & Puccio, L. (eds), *Wavelets: Theory, Algorithms and Applications.*

Witten, I. H. & Bell, T. C. 1991. The Zero Frequency problem: Estimating the Probabilities of Novel Events in Adaptive Text Compression. *IEEE Transactions on Information Theory*, **37**(July), 1085–1094.

Wu, R.-S. & Wang, Y. 1999 (July). New Flexible Segmentation Technique in Seismic Data Compression Using Local Cosine Transform. *Pages 784–794 of: Wavelet Applications in Signal and Image Processing VII*, vol. 3813.

Yagel, R. & Shi, Z. 1993 (October). Acelerating Volume Animation by Space–Leaping. *Pages 62–69 of: Proceedings of Visualization'93.*

Yeo, B. L. & Liu, B. 1995. Volume Rendering of DCT-based Compressed 3D Scalar Data. *IEEE Transactions on Visualization and Computer Graphics.*, **1**(1), 29–43.

Ziv, J. & Lempel, J. 1977. A Universal Algorithm for Sequential Data Compression. *IEEE Transaction on Information Theory*, **23**(3), 337–343.