Visualização Volumétrica em Multiresolução

Anselmo C. de Paiva
paiva@tecgraf.puc-rio.br
TecGraf-Departamento de Informática-PUC-Rio
paiva@ufma.br
Departamento de Informática
Universidade Federal do Maranhão - UFMA

Jonas Gomes
jonas@impa.br
Instituto de Matemática Pura e Aplicada - IMPA

PUC-RioInf.MCC04/99 January, 1999

ABSTRACT: A volumetric data set is a collection of data in which each data has an associated location in the three-dimensional space. In this work we review the wavelets theory and discuss the application of this technique to the visualization of volumetric data. We show the main algorithms, and discuss some implementation issues.

Keywords: computer graphics, volume rendering, ray casting, medical imaging, wavelets, multiresolution.

Resumo: Um conjunto de dados volumétricos é uma coleção de dados amostrados em uma grade no espaço 3D. Neste trabalho apresentamos uma introdução da teoria de wavelets e discutimos a aplicação desta técnica à visualização de dados volumétricos. Apresentamos os principais algoritmos e discutimos alguns aspectos de sua implementação.

Palavras-chave: computação gráfica, visualização de volumes, imagens médicas, wavelets, multiresolução.

Índice

Capítulo 1

Introdução

Visualização é um termo empregado para descrever os métodos que permitem a extração de informações relevantes a partir de complexos conjuntos de dados, o que é feito com a utilização de técnicas de computação gráfica e processamento de imagens. Uma subárea importante é representada pelos métodos de Visualização Volumétrica.

Segundo [Mccormick et al., 1987], visualização volumétrica é o conjunto de técnicas utilizadas na visualização de dados associados a regiões de um volume, tendo como principal objetivo a exibição de seu interior para explorar sua estrutura e facilitar sua compreensão. De uma maneira geral, podemos definir visualização volumétrica como o processo realizado com o objetivo de obter uma melhor compreensão da estrutura contida nos dados volumétricos. Entendemos como dados volumétricos, um conjunto de valores escalares amostrados em uma grade tridimensional.

Em algumas aplicações de visualização volumétrica, em geral associadas a ambientes distribuídos, WWW e realidade virtual, existe uma demanda por representações comprimidas dos dados e a possibilidade de extração de versões em várias resoluções deste dado. Esta demanda é resultante das necessidades de adequação ao meio de comunicação, gerando a exibição mais apropriada em taxas interativas. Como resultado disto, vários modelos de multiresolução foram propostos na literatura para representação de objetos tridimensionais. Por outro lado, os volumes de dados em geral possuem a característica de alto consumo de memória, o que influencia os requisitos de armazenamento e de tempo de visualização. Isto acaba resultando também na necessidade de adoção de modelos aproximados do conjunto de dados.

Para atender estes dois requisitos uma escolha natural são os modelos de representação dos dados em multiresolução. Entre as vantagens destes modelos, podemos destacar: redução da quantidade de dados, quando representações em baixa resolução são suficientes; acesso rápido aos dados, através

da exploração da indexação espacial inerente aos modelos; processamento hierárquico.

A abordagem principal para modelagem em multiresolução é baseada em um modelo de decomposição, onde a resolução é controlada pelo nível de refinamento da subdivisão do domínio. A idéia básica deste modelo é que a precisão da representação é proporcional ao número de células de subdivisão.

Diferentes arquiteturas foram propostas para lidar com a representação em multiresolução. Estas arquiteturas vão desde uma simples coleção de várias versões do objeto em diferentes resoluções (por exemplo, nós LOD (*level of detail*) do Open Inventor ou VRML [Wernecke, 1994]); passando por modelos que mantém relações entre dois níveis de detalhes consecutivos (por exemplo *quadtrees*, representações piramidais ou hierárquicas); até modelos mais sofisticados que mantém uma estrutura compacta, de onde podem ser extraídas diferentes representações dos dados, geradas com resolução diferente. Uma arquitetura ainda mais flexível consiste em considerar uma representação cuja resolução é variável espacialmente. Representações deste tipo, em geral, requerem o uso de métodos não-lineares e são difíceis de serem obtidas.

Podemos classificar as técnicas que tratam multiresolução de dados volumétricos em duas grandes áreas: baseados em superfícies ou em representações volumétricas.

Na primeira classe se encontram os métodos que obtém uma representação por superfícies do dado volumétrico, representação aproximada. Como representante desta classe temos o algoritmo proposto por [Williams, 1993], que é baseado no uso de uma partição hierárquica recursiva (parecida com uma (*octree*), e a proposta de [Cignoni and Scopigno, 1995] baseada em decomposições tetraédricas.

Na outra classe podemos incluir as técnicas diretas, que representam os dados volumétricos em multiresolução sem a conversão para uma representação por superfícies, o que garante uma eficiente representação de objetos amorfos. Nesta classe encontram-se extensões para 3D dos métodos de multiresolução aplicados a imagens (pirâmides de imagens).

As representações volumétricas em multiresolução desenvolvidas para dados volumétricos, são em geral baseadas na transformada de *wavelet*. Em [Muraki, 1992] e [Muraki, 1993] estas tansformadas foram aplicadas a dados volumétricos com o objetivo de obter aproximações compactas. Enquanto Westerman [Westerman, 1994] demonstrou como podemos utilizar o algoritmo de *ray casting* diretamente sobre os coeficientes de *wavelet* que representam o volume.

Neste trabalho apresentaremos as técnicas de visualização de volumes em multiresolução baseadas na transformada de *wavelets*. No capítulo seguinte

é apresentada uma introdução à transformada de *wavelets* com sua extensão para 2 e 3 dimensões. No capítulo 3 é apresentada a teoria básica de visualização volumétrica, e são descritos os algoritmos *ray casting* e *splatting*. Em seguida, capítulo 4, duas técnicas para visualização de volumes baseadas nos coeficientes de sua representação na base de *wavelets* são apresentadas.

Capítulo 2

Transformada de *Wavelets*

2.1 Introdução

A decomposição de sinais em bandas de frequência tem se mostrado uma aplicação bastante útil para tratamentos de sinais. Esta decomposição fornece uma representação intermediária entre o espaço temporal e o espaço das frequências de Fourier.

A transformada de Fourier (FT) de uma função $f(x)$ fornece uma medida das frequências do sinal, mas estas informações não estão localizadas no espaço. Isto se dá em razão do átomo tempo-frequência da transformada, e^{-iwt}, não possuir localização temporal. Para localizar melhor a informação, foi introduzida por Gabor [Gabor, 1946] uma transformada de Fourier com janela(WFT), através da introdução de uma "janela" espacial $g(x)$ à integral da Transformada de Fourier. Assim, a Transformada de Fourier com Janela mede a amplitude de uma componente senoidal de frequência que ocorre na vizinhança de um ponto u. O problema desta representação é a escala fixa, introduzida pela janela, cobrindo o domínio tempo-frequência. Surge então a necessidade de um filtro espacial que se adapte às irregularidades do sinal, o que propõe a Transformada de *Wavelets* (WT). Esta transformada surgiu como resposta às deficiências da Transformada de Fourier e Transformada de Fourier com Janela, através da introdução de uma base de funções de suporte compacto, e que variam sua posição e frequência. Estas funções são denominadas *wavelet* (ou *ondelettes* em francês) (figura 2.1).

Segundo [Castleman, 1996] a transformada de *wavelets* pode ser compreendida como uma representação análoga a de uma pauta musical. Considere a pauta apresentada na figura 2.2, e que pode ser vista como uma representação bidimensional do espaço tempo-frequência. Frequência (quão grave ou agudo é o som) aumenta da base da escala para seu topo, enquanto o

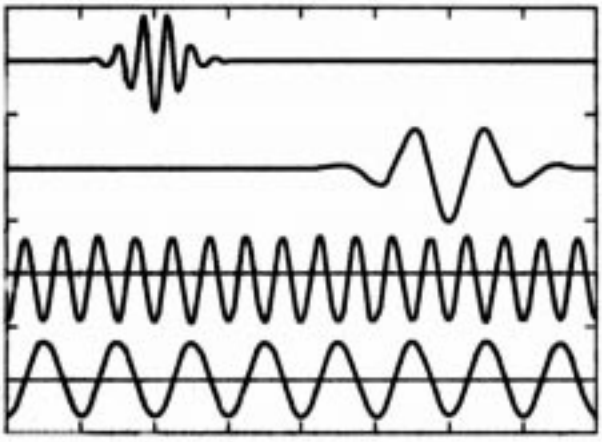

Figura 2.1: Ondas (waves) e wavelets.

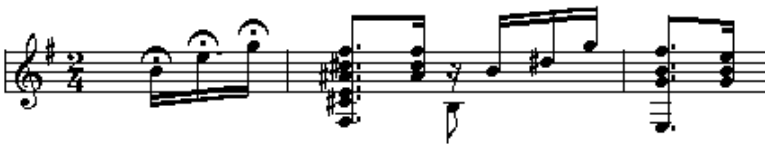

Figura 2.2: Notação Musical como um espaço tempo-frequência.

tempo (medido em batidas (compassos)) avança para a direita. Cada nota corresponde a uma componente de *wavelet* que aparecerá durante a execução da música. A duração de cada *wavelet* é codificada pelo tipo de nota (e.g., colcheia, semi colcheia, etc.).

Se fôssemos analizar uma música gravada e gerar a partir da sua execução uma pauta, estaríamos aplicando uma transformada de *wavelet* à música (sinal sonoro). De maneira semelhante um músico executando uma música a partir de sua pauta, pode ser visto como executando uma transformada de *wavelet* inversa, pois ele reconstrói o sinal sonoro a partir de uma representação no domínio tempo-frequência.

A idéia básica da transformada de *wavelet* é a decomposição de uma função em uma base que gera um subespaço de funções, de modo que esta representação possua propriedades bem determinadas. Por exemplo, localizar no tempo as frequências.

Uma *wavelet* é uma função, através da qual, por translações e escalas, formamos uma base do subespaço desejado. Chamamos ϕ a função de escala que permite gerar uma versão filtrada por passa-baixa do sinal e ψ a função

de *wavelet* (*wavelet* mãe) que permite a obtenção dos detalhes entre dois filtros ϕ em escalas consecutivas.

A análise de *wavelets* adota um função base e realiza suas análises variando temporalmente o suporte da base, adaptando-se de modo automático nas regiões em que a função apresenta altas frequências.

A Transformada Contínua de Wavelet (CWT) foi introduzida por Grossman e Morlet [Grossman and Morlet, 1984]. Se $\psi(x)$ é uma função real cujo espectro de Fourier, $\hat{\psi}$, satisfaz o critério de admissibilidade ([Chui, 1992] e [Grossman and Morlet, 1984]):

$$C_\psi = \int_{-\infty}^{+\infty} \frac{\mid \hat{\psi(u)} \mid^2}{\mid s \mid} ds < \infty, \tag{2.1}$$

então $\psi(x)$ é denominada uma *wavelet* básica. Devido à variável s no denominador é necessário que:

$$\hat{\psi}(0) = 0 \Rightarrow \int_{-\infty}^{+\infty} \psi(x)\, dx = 0 \tag{2.2}$$

A equação 2.2 força que a função ψ oscile de modo a cancelar áreas na região positiva com áreas na região negativa do espaço de modo a anular a integral. Para isto esta função deve possuir característica ondulatória, com seu gráfico formando uma onda de curta duração e frequência crescente, conforme mostrado na figura 2.1.

Uma família de *wavelets* $\{\psi_{s,t}(x)\}$ é gerada através de escalas e translações da *wavelet* básica, $\psi(x)$:

$$\psi_{s,t}(x) = \frac{1}{\sqrt{s}} \psi \left(\frac{x \Leftrightarrow t}{s} \right), \tag{2.3}$$

com $s > 0$ e $t, s \in \Re$. A variável s representa a escala (largura) de uma função da base, enquanto t representa a translação da função ao longo do eixo x.

Exemplo 1

Considere a função, com gráfico apresentado na figura 2.3, que é dada por:

$$\psi(x) = \begin{cases} 1, & \text{se } x \in \left[0, \frac{1}{2}\right) \\ \Leftrightarrow 1, & \text{se } x \in \left[\frac{1}{2}, 1\right) \\ 0, & \text{se } x < 0 \text{ ou } x > 1 \end{cases} \tag{2.4}$$

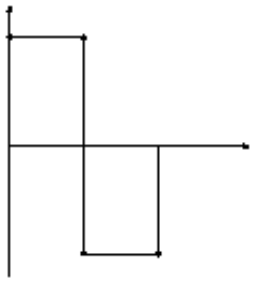

Figura 2.3: *Wavelet* de Haar

Esta função satisfaz a condição de admissibilidade (equação 2.1), sendo possível mostrar que o conjunto $\psi_{m,n}$, onde:

$$\psi_{m,n}(u) = 2^{-\frac{m}{2}}\psi(2^{-m}u \Leftrightarrow n), \quad m, n \in Z, \tag{2.5}$$

forma uma base ortonormal de $L^2(\Re)$, gerando pois uma base de wavelets. Para demonstração deste fato veja [Daubechies, 1992]. Esta é a mais simples e antiga base de wavelets, tendo sido proposta por Haar, no início do século com o nome de transformada de Haar, e muito utilizada em processamento de imagens.

Exemplo 2

Considere a distribuição Gaussiana com média 0 e variância 1, dada por:

$$\phi(u) = \frac{1}{\sqrt{2\pi}}e^{-\frac{u^2}{2}}, \tag{2.6}$$

e sua segunda derivada dada por:

$$\psi(u) = \frac{1}{\sqrt{2\pi}}(u^2 \Leftrightarrow 1)e^{-\frac{u^2}{2}} \tag{2.7}$$

Esta função, conhecida como sombrero, também satisfaz a condição de admissibilidade e define uma transformada de wavelets. A figura 2.4 apresenta o gráfico destas duas funções.

O fator de escala $\frac{1}{\sqrt{s}}$ na equação 2.3 assegura que a norma das funções da base de *wavelet* são todas iguais, pois:

$$\left\| f\left(\frac{x \Leftrightarrow t}{s}\right)\right\| = \sqrt{\int_{-\infty}^{\infty} \mid f\left(\frac{x \Leftrightarrow t}{s}\right) \mid^2 \, dx} = \sqrt{s} \parallel f(x) \parallel, \tag{2.8}$$

assim todas as funções da base de *wavelets* possuem média zero, como a *wavelet* básica.

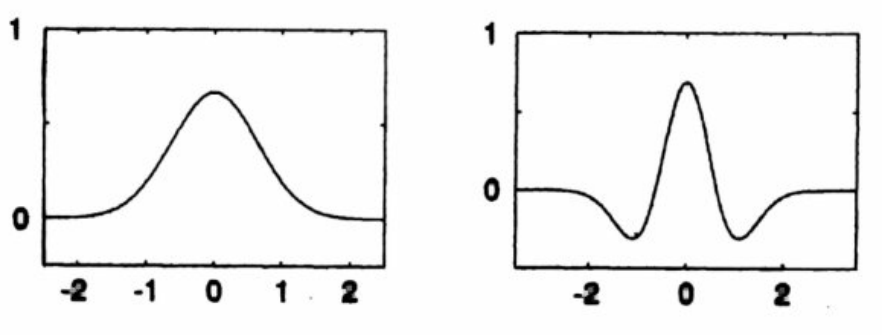

Figura 2.4: Gráfico da Gaussiana e da wavelet sombrero

A transformada contínua de $f(x)$ em relação à *wavelet* $\psi(x)$ é:

$$W_f(s,t) = \langle f, \psi_{s,t} \rangle = \int_{-\infty}^{+\infty} f(x)\psi_{s,t}(x)\,dx \tag{2.9}$$

Grossman e Morlet [Grossman and Morlet, 1984] mostraram que a transformada contínua inversa de *wavelet* é dada por:

$$f(x) = \frac{1}{C_\psi} \int_0^\infty \int_{-\infty}^\infty W_f(s,t)\psi^*_{s,t}(x)\,dt\,\frac{ds}{s^2} \tag{2.10}$$

2.1.1 Transformada Discreta de *Wavelets*

Para implementar a transformada de *wavelets* é necessário encontrar uma maneira apropriada de realizar a amostragem dos parâmetros s e t, de modo a obter um conjunto de *wavelets* nestes parâmetros discretos. O problema central neste desenvolvimento é a discretização do domínio tempo-escala onde a transformada é definida.

Uma característica marcante da transformada de *wavelets* é sua invariância quanto a mudanças de escala. Ou seja, se aplicarmos uma mudança de escala a uma função f e simultâneamente aplicarmos a mesma mudança de escala ao espaço em que esta função é descrita, não causaremos nenhuma mudança na transformada de *wavelet* de f (W_f). Formalmente, se tomarmos $f_{s_0}(t) = s_o^{-\frac{1}{2}} f\left(\frac{t}{s_o}\right)$ então:

$$W_{f_{s_0}}(s_0 s, s_0 t) = W_f(s,t) \tag{2.11}$$

A discretização da transformada deve manter esta propriedade. Para isto, ao passarmos da escala $s_m = s_0^m$ para a escala $s_{m+1} = s_0^{m+1}$, devemos

incrementar o tempo por um fator de escala s_0. Se escolhermos um intervalo de tempo t_0 e tomarmos os intervalos de amostragem no tempo como $\Delta t = s_0^m t_0$, geramos um reticulado de amostragem no tempo, representado por:

$$t_{m,n} = n s_0^m t_0, \quad n \in Z, \tag{2.12}$$

e o reticulado do espaço tempo-escala fica definido por:

$$\Delta_{s_o,t_0} = \{(s_0^m, n s_0^m t_0);\ m, n \in Z\} \tag{2.13}$$

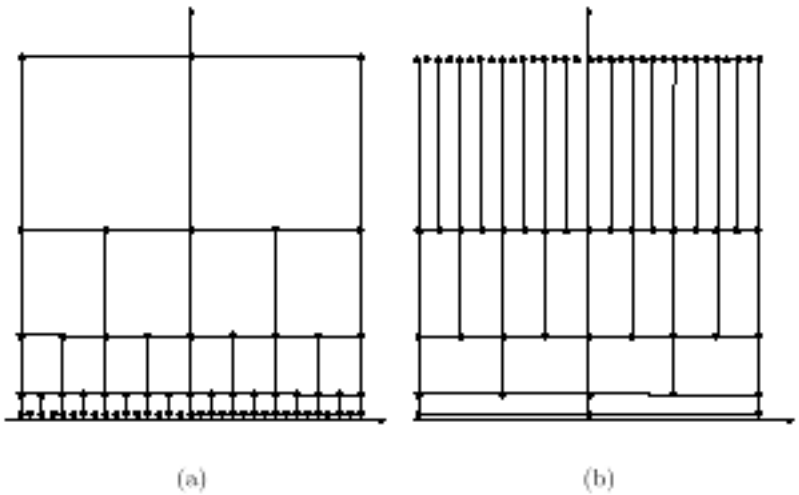

Figura 2.5: (a) Reticulado tempo-escala (b) Reticulado tempo-frequência

Um dos reticulados mais usados é o baseado na partição diádica do espaço. Considerando $s_0 = 2$, formamos o reticulado diádico, representado na figura 2.5a, e expresso por:

$$\Delta_{2,t_0} = \{(2^m, n 2^m t_0);\ m, n \in Z\} \tag{2.14}$$

Considerando que a frequência é o inverso da escala, podemos obter o reticulado de discretização no espaço tempo-frequência, para uma frequência inicial w_0, (figura 2.5 b), por:

$$\Delta_{2w_0,t_0} = \{(2^{-m} w_0, n 2^{-m} t_0);\ m, n \in Z\} \tag{2.15}$$

O reticulado gerado no espaço tempo-frequência divide este espaço em regiões retangulares de mesma área, conforme mostrado na figura 2.5 (b).

A discretização da transformada de *wavelet* $W_{f_{m,n}} = \langle f, \psi_{s,t}(u) \rangle$ no espaço tempo-escala é dada por:

$$W_{f_{m,n}} = \langle f, \psi_{m,n}(u) \rangle \tag{2.16}$$

onde

$$\begin{aligned} \psi_{m,n}(u) &= \psi_{s_0^m, nt_o s_o^m}(u) \\ &= s_0^{-m/2} \psi \left(\frac{u \Leftrightarrow nt_0 s_0^m}{s_0^m} \right) \\ &= s_0^{-m/2} \psi(s_0^{-m} u \Leftrightarrow nt_0). \end{aligned} \tag{2.17}$$

2.1.2 Noção Intuitiva

Considere a *wavelet* sombrero figura 2.4, dada por :

$$g(x) = \frac{1}{\sqrt{2\pi}} (x^2 \Leftrightarrow 1) e^{-\frac{x^2}{2}}, \tag{2.18}$$

Se usarmos esta função como *wavelet* básica (ψ), então a função a ser decomposta deve ser expressa como uma combinação linear de diferentes versões escaladas $g(x \Leftrightarrow t)$ e transladadas $g(x/s)$ desta *wavelet*.

Assim, a *wavelet* transladada e escalada pode ser escrita como:

$$\psi_{s,t}(x) = g \left(\frac{x \Leftrightarrow t}{s} \right), \tag{2.19}$$

onde s e t representam a escala e translação da *wavelet* g.

Para um conjunto de amostras representando uma dada função $f(x)$, a decomposição de *wavelet* pode ser escrita como:

$$f(x) = \sum_{i=0}^{S} \sum_{j=0}^{T} c_{i,j} w_{i,j}(x), \tag{2.20}$$

que é uma combinação linear das funções $\psi_{i,j}$, onde $c_{i,j}$ são os coeficientes de *wavelet*, e seus valores representam a contribuição relativa de uma *wavelet* respectiva na decomposição. Assim, verificamos que uma decomposição em *wavelet* de uma função f é uma função bidimensional em s e t.

A transformada de *wavelet* desta função discreta pode ser calculada com o auxílio de uma regra de quadratura de modo, a ser expressa por:

$$W_{f(x)}(s,t) = \frac{1}{\sqrt{s}} \sum_{k=0}^{N} f(k) \psi_{(s,t)}(k) \tag{2.21}$$

Para implementar a transformada discreta de *wavelet* basta escolher a discretização apropriada e aplicar a equação 2.21. No algoritmo 2.1 utilizamos uma discretização arbitrária.

Algoritmo 2.1

```
** Implementação direta da transformada de wavelets
void WaveTrans1D ( float *data, float wt[][], int n)
{
   wt[0][0] = 0.0;
   s = a0;
   for (i=0; i < n; i++) {
      sqa = sqrt(s);
      t = s * 0.99
      for (j=0; j < n; j++) {
         sum = 0.0;
         for (k=0; k < n; k++){
            sum += data[k] * mother(s, t, k);
            wt[i][j] = sum /sqa;
            t *= 1.03; ! particao arbitraria dos tempo
         }
         s *= 1.01; ! particao arbitraria dos espaço
   }
}
float mother ( float a, float b, float t )
{
   float y, z;
   z = sqrt(a);
   if ( z != 0.0 ) z = 1.0 / z;
   else z = 0.0;
   return z*sen(4(t+b)/a)*exp(-(t+b)/a*((t+b)/a));
}
```

2.2 Análise de Multiresolução

Um problema bastante difícil consiste em determinar a *wavelet* mãe ψ associada a uma determinada função escala ϕ. Uma solução para esse problema pode ser obtida usando análise de multiresolução. A análise de multiresolução é uma outra forma de introduzir a transformada de wavelets. Formalmente, a Análise de Multiresolução está relacionada à descrição das funções f de L^2 como um limite de aproximações sucessivas, sendo cada aproximação uma

versão suave de f. As aproximações sucessivas utilizam diferentes resoluções, de onde se origina a denominação.

Mais precisamente, uma análise de multiresolução é constituida por um conjunto de espaços aninhados.

$$V_j \subset L^2(\Re), m \in Z$$
$$\cdots \subset V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2} \subset \cdots \tag{2.22}$$

tal que,

1. $V_j \subset V_{j-1}$
2. $\bigcap_{j \in Z} V_j = \{0\}$
3. $\overline{\bigcup}_{j \in Z} V_j = L^2(\Re)$
4. $f \in V_j \Leftrightarrow f(2u) \subset V_{j-1}$
5. Existe uma função $\phi \in V_0$ tal que o conjunto $\{\phi(u \Leftrightarrow k);\ k \in Z\}$ forma uma base ortonormal de V_0. Como consequência $\phi_{jk} = 2^{-\frac{j}{2}}\phi(2^{-j}x \Leftrightarrow k), k \in Z$ é base ortonormal de V_j.

1, 2 e 3 asseguram que qualquer função $f \in L^2(\Re)$ pode ser descrita pela análise de multiresolução. 4 assegura que os espaços V_m correspondem a diferentes escalas, e de 5 vem que os espaços V_m são invariantes a translações no tempo. Podemos notar que a análise de multiresolução aproxima uma função f por uma função f^j em cada subespaço V_j. Como a união de todos os V_j é densa em $L^2(\Re)$, temos que a aproximação converge para a função original:

$$f = \lim_{j \to +\infty} f^j \tag{2.23}$$

Na figura 2.6 observamos a função f e suas representações nos espaços de escala V_0 e V_{-1} da representação em multiresolução de Haar.

A projeção ortogonal de uma função $f \in L^2(\Re)$ em V_j é obtida usando uma operação de filtragem de f com diferentes núcleos $\phi_{j,k}$, $k \in Z$ que definem um filtro passa-baixa. Indicando a frequência de corte destes filtros por α_j (figura 2.7), podemos concluir que o espaço V_j é constituído por funções cujas frequências estão no intervalo $[\Leftrightarrow\alpha_j, \alpha_j], \alpha_j > 0$.

Uma outra interpretação da equação 2.23 pode ser obtida analisando a figura 2.8. Nesta figura podemos perceber que o espaço V_{j-1} é obtido do espaço V_j adicionando todas as funções de $L^2(\Re)$ com frequências na banda

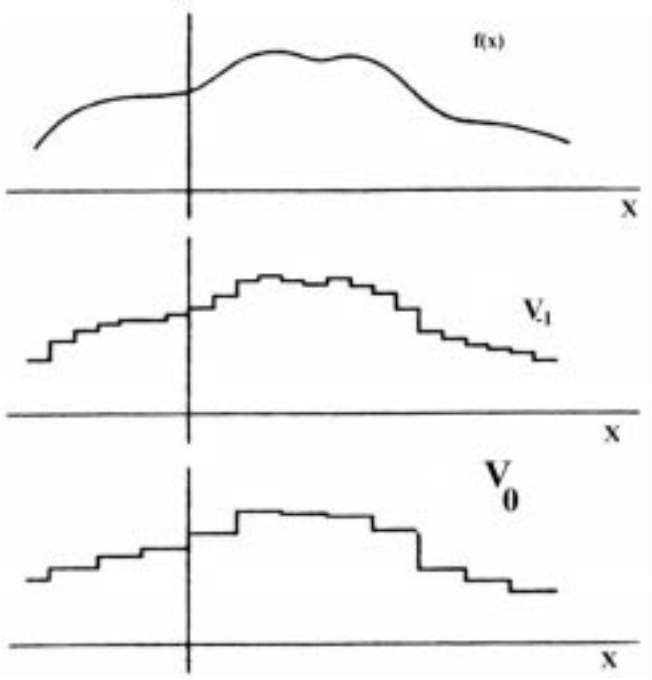

Figura 2.6: Aproximações de uma função em duas escalas [Daubechies, 1992].

$[\Leftrightarrow\alpha_j, \alpha_j]$ do espectro. Indicamos este "espaço de detalhes" por W_j, que é ortogonal a V_j, assim temos:

$$V_{j-1} = V_j \oplus W_j \tag{2.24}$$

Da equação 2.24, verificamos que para cada $j \in Z$, teremos W_j como o complemento ortogonal e, se fixarmos $J_0 \in Z$, para cada $j < J_0$ teremos (figura 2.8):

$$V_j = V_{J_0} \oplus \bigoplus_{k=0}^{J_0-j} W_{J_0-k} \tag{2.25}$$

2.2.1 Função de Escala e Espaços V_j

Se $\phi \in V_0 \subset V_{-1}$, então existe uma sequência $\{h_k\} \in l^2(\Re)$ de modo que:

$$\phi(x) = 2\sum_k h_k \phi(2x \Leftrightarrow k), \tag{2.26}$$

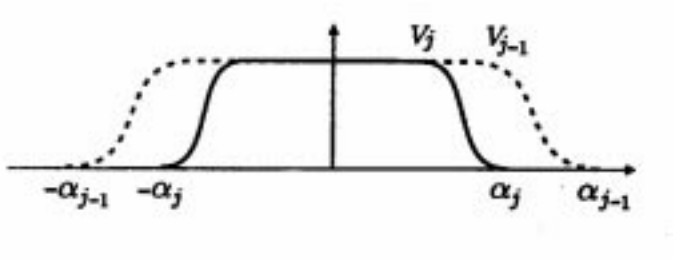

Figura 2.7: Espectros das funções de escala.

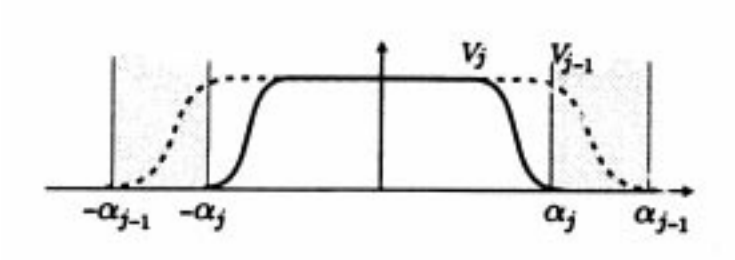

Figura 2.8: Bandas de Frequência entre V_j e V_{j-1} [Gomes and Velho, 1998a].

onde

$$h_k = \langle \phi, \phi_{-1,k} \rangle, \text{ e } \sum_{k \in Z} \|h_k\|^2 = 1 \tag{2.27}$$

A equação 2.26 é denominada equação de escala dupla ou equação de refinamento, e a coleção de funções $\{\phi(x \Leftrightarrow t)\}$ forma uma base de ortonormal de V_j.

Assim, a função de escala é definida pela equação de escala dupla e pela normalização ([Daubechies, 1988] e [Daubechies and Lagarias, 1991]):

$$\int_{-\infty}^{\infty} \phi(x)\, dx = 1 \Longleftrightarrow \hat{\phi}(0) = 1 \tag{2.28}$$

Na maioria dos casos a expressão de ϕ não está disponível, no entanto existem algoritmos rápidos para avaliar ϕ nos pontos do reticulado diádico $\{x = 2^{-j}k, k \in Z\}$, o algoritmo cascade. Podemos verificar que na maioria das aplicações não se trabalha com ϕ, mas somente com os coeficientes h_k.

Da equação 2.26 temos que a transformada de Fourier de ϕ satisfaz

$$\hat{\phi}(w) = H(w/2)\hat{\phi}(w/2), \tag{2.29}$$

onde H é uma função 2π-periódica definida por:

$$H(w) = \sum_k h_k e^{-ikw} \tag{2.30}$$

2.2.2 Função de *Wavelet* e Espaço W_j

W_j é o espaço complementar a V_j em relação a V_{j-1}:

$$V_{j-1} = V_j \oplus Wj, \tag{2.31}$$

este espaço contém a informação de diferença entre uma aproximação a escala j e uma aproximação a escala $j \Leftrightarrow 1$. Como consequência temos que:

$$\bigoplus W_j = L^2(\Re), \tag{2.32}$$

o que mostra que W_j não é único. O espaço W_0 é gerado por uma *wavelet* ψ. Mais precisamente, existe uma *wavelet* ψ que satisfaz à propriedade de admissibilidade, e tal que $\psi(x \Leftrightarrow k)$ é base de W_0. Como ψ é um elemento de V_{-1}, então existe uma sequência $g_k \in l^2(Z)$, tal que

$$\psi(x) = 2\sum_k g_k \phi(2x \Leftrightarrow k) \tag{2.33}$$

Aplicando a transformada de Fourier à equação 2.33 temos:

$$\hat{\psi}(w) = G(w/2)\hat{\phi}(w/2), \tag{2.34}$$

onde G é uma função 2π periódica

$$G(w) = \sum_k g_k e^{-ikw}, \tag{2.35}$$

donde tiramos que

$$\sum_k g_k = 0 \quad \text{ou} \quad G(0) = 0 \tag{2.36}$$

2.3 A Transformada Rápida de *Wavelets*

Mallat [Mallat, 1989] definiu um algoritmo para a transformada discreta de *wavelet* que é mais eficiente que o cálculo dos produtos internos representados

na equação 2.9 para o caso contínuo. O algoritmo apresentado pode ser deduzido através da análise de multiresolução descrita na seção anterior.

Assumindo que temos uma função f_j que pertence a um dado espaço V_j, podemos descrever esta função, através da análise de multiresolução, por:

$$f_j = f_{j+1} + o_{j+1}, \tag{2.37}$$

onde f_{j+1} e o_{j+1} são as projeções de f nos espaços V_{j+1} e W_{j+1} respectivamente.

Aplicando 2.37 recursivamente, obtemos:

$$\begin{aligned} f_j &= f_{j+1} + o_{j+2} + o_{j+1} \\ f_j &= f_{j+N} + o_{j+N} + \cdots + o_{j+2} + P_{W_{j+1}} f \end{aligned} \tag{2.38}$$

A equação 2.38 descreve a função f_j através de suas projeções no espaços $W_{j+1} \cdots W_{j+N}$, e de um resíduo no espaço de escala V_{j+1}. Este processo recursivo está ilustrado na figura 2.9.

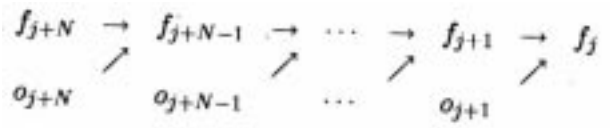

Figura 2.9: Reconstrução de uma Função na Base de *Wavelets*.

Posto que $W_j \subset V_{j-1}$ e $V_j \subset V_{j-1}$, podemos reconstruir a função original através de suas projeções. O processo de reconstrução, apresentado na figura 2.10, é o reverso do apresentado na figura 2.9.

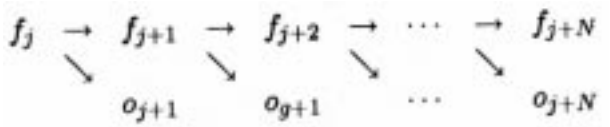

Figura 2.10: Decomposição em *Wavelets* de uma Função.

Como consequência da análise de multiresolução, temos que os conjuntos $\{\phi_{jn};\ n \in Z\}$ e $\{\psi_{jn};\ n \in Z\}$, definidos por:

$$\begin{aligned} \phi_{jk}(x) &= 2^{-\frac{j}{2}} \phi(2^{-j}x \Leftrightarrow k) \\ \psi_{jk}(x) &= 2^{-\frac{j}{2}} \psi(2^{-j}x \Leftrightarrow k), \end{aligned} \tag{2.39}$$

formam uma base ortonormal de V_j e W_j.

Assim, os operadores P_{V_j} e P_{W_j}, usados para obter f_j e o_j podem ser definidos através de produtos internos com os elementos destas bases.

$$\begin{aligned} P_{V_j} &= \sum_n \langle f, \phi_{jn} \rangle \phi_{jn} = \sum_n \left(\int f(x) \overline{\phi_{jn}(x)\, dx} \right) \phi_{jn} \\ P_{W_j} &= \sum_n \langle f, \psi_{jn} \rangle \psi_{jn} = \sum_n \left(\int f(x) \overline{\psi_{jn}(x)\, dx} \right) \phi_{jn}, \end{aligned}$$

onde $\langle f, \phi_{jn} \rangle$ e $\langle f, \psi_{jn} \rangle$ são os coeficientes de escala c^j e de detalhes d^j. Estes operadores ficam completamente definidos por:

$$P_{V_j} = \sum_n c_n^j \phi_{jn} \tag{2.40}$$

$$P_{W_j} = \sum_n d_n^j \psi_{jn} \tag{2.41}$$

O processo recursivo de decomposição/reconstrução utiliza somente projeções entre subespaços consecutivos da estrutura de multiresolução. Assim, podemos utilizar a relação de escala dupla (equação 2.26), que relaciona as funções da base de escala e de *wavelet*, V_j e W_j no nível j, com o espaço de escala subsequente no nível $j \Leftrightarrow 1$.

$$\phi(x) = \sum_k h_k \phi_{-1,k}(x) \tag{2.42}$$

$$\psi(x) = \sum_k g_k \psi_{-1,k}(x) \tag{2.43}$$

Substituindo 2.39 em 2.42 temos:

$$\begin{aligned} \phi_{jk}(x) &= 2^{-\frac{j}{2}} \phi(2^{-j} x \Leftrightarrow k) = 2^{-\frac{j}{2}} \sum_n h_n \phi_{-1,k}(2x \Leftrightarrow k) \\ &= 2^{-\frac{j}{2}} \sum_n h_n 2^{1/2} \phi(2^{-j+1} x \Leftrightarrow 2k \Leftrightarrow n) = \sum_n h_n \phi_{j-1,2k+n}(x) \\ &= \sum_n h_{n-2k} \phi_{j-1,n}(x), \end{aligned} \tag{2.44}$$

e similarmente pra ψ

$$\begin{aligned}
\psi_{jk}(x) &= 2^{-\frac{j}{2}}\psi(2^{-j}x \Leftrightarrow k) \\
&= 2^{-\frac{j}{2}}\sum_n g_n \phi_{-1,k}(2x \Leftrightarrow k) \\
&= 2^{-\frac{j}{2}}\sum_n g_n 2^{1/2}\phi(2^{-j+1}x \Leftrightarrow 2k \Leftrightarrow n) \\
&= \sum_n g_n \phi_{j-1,2k+n}(x) \\
&= \sum_n g_{n-2k}\phi_{j-1,n}(x) \qquad (2.45)
\end{aligned}$$

Se substituirmos as equações 2.42 no produto interno que define c^j e d^j, obtemos uma maneira de calcular estes coeficientes a partir das sequências h_k e g_k.

$$\begin{aligned}
c_k^j &= \langle f_j, \phi_{j,k}\rangle \\
&= \langle f_j, \sum_n h_{n-2k}\phi_{j-1,k}\rangle \\
&= \sum_n h_{n-2k}\langle f_j, \phi_{j-1,k}\rangle \qquad (2.46)
\end{aligned}$$

$$\begin{aligned}
d^j &= \langle f_j, \psi_{j,k}\rangle \\
&= \langle f_j, \sum_n g_{n-2k}\phi_{j-1,k}\rangle \\
&= \sum_n g_{n-2k}\langle f_j, \phi_{j-1,k}\rangle \qquad (2.47)
\end{aligned}$$

Como $\langle f, \phi_{j-1,k}\rangle = c^{j-1}$, temos

$$c_k^j = \sum_n h_{n-2k}c_n^{j-1} \qquad (2.48)$$

$$d_k^j = \sum_n g_{n-2k}c_n^{j-1} \qquad (2.49)$$

Como assumimos que a função a ser decomposta possui uma escala natural associada a ela, dizemos que a função f a ser decomposta pertence ao espaço de escala V_0, sendo descrita pela sequência (c_n^0) contendo 2^J coeficientes. Em razão da existência do fator $2k$ no índice do filtro, verificamos que apenas os coeficientes pares são mantidos para o próximo passo da

transformação. Isto pode ser visto na figura 2.11, onde iniciamos com um vetor com $n = 8 = 2^3$ elementos ($J = 3$), que é decomposto nas sequências $(d^1_{n/2}), (d^1_{n/4}), \cdots (d^1_{n/2^J})$, e $(c^J_{n/2^J})$, resultando em uma sequência com o mesmo número de elementos da sequência inicial.

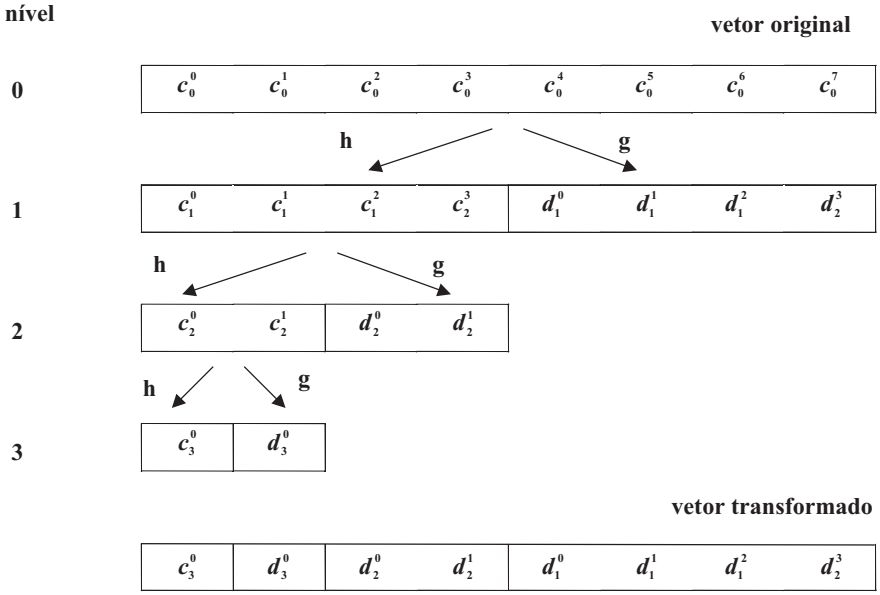

Figura 2.11: Exemplo de Aplicação da FWT

O processo de reconstrução gera os coeficientes da sequência que representa a função na escala desejada, a partir dos coeficientes de *wavelet* (c^j e d^j). A reconstrução exata é possível em razão da operação realizada representar apenas uma transformação em bases ortogonais.

Podemos verificar que no processo de decomposição a função f_j é dividida em suas componentes, através dos operadores $P_{V_{j+1}}$ e $P_{W_{j+1}}$.

$$\begin{aligned} f_j &= P_{V_{j+1}} f + P_{W_{j+1}} f \\ &= \sum_k c_k^{j+1} \phi_{j+1,k}(x) + \sum_k d_k^{j+1} \psi_{j+1,k} \end{aligned} \tag{2.50}$$

No processo de reconstrução o objetivo é descobrir os coeficientes c^j_n a partir dos coeficientes c^j e d^j, conforme mostra a figura 2.12 onde a reconstrução de uma sequência com 2^3 elementos é realizada. Como $c^j_n = \langle f_j, \phi_{j,n} \rangle$

substituindo 2.50, podemos escrever:

$$\begin{aligned} c_n^{j-1} &= \langle \sum_k c_k^{j+1}\phi_{j+1,k}(x) + \sum_k d_k^{j+1}\psi_{j+1,k}, \phi_{j-1,n}\rangle \\ &= \sum_k c_k^{j+1}\langle \phi_{j+1,k}(x), \phi_{j,k}\rangle + \sum_k d_k^{j+1}\langle \psi_{j+1,k}, \phi_{j,n}\rangle \end{aligned} \tag{2.51}$$

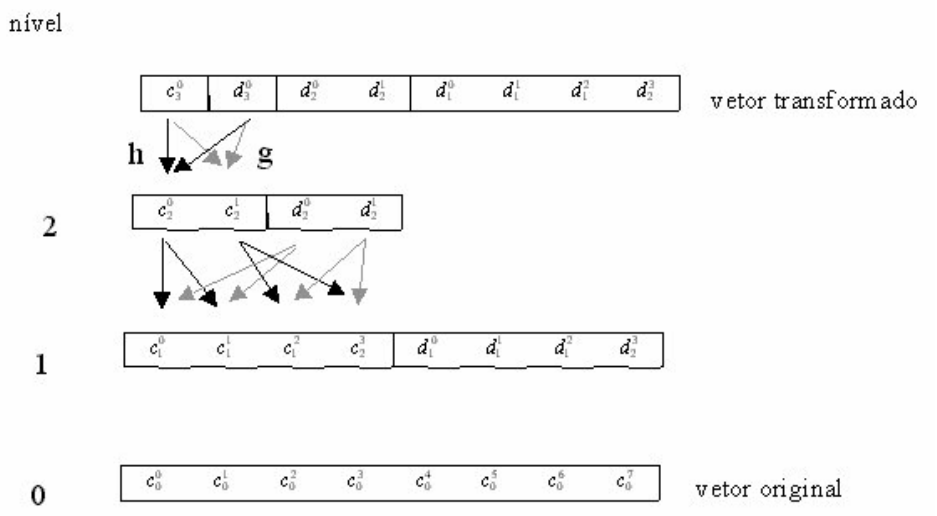

Figura 2.12: Exemplo de Aplicação da Inversa WT

Da equação 2.27 verificamos que:

$$h_n = \langle \phi_0, \phi_{-1,n}\rangle \tag{2.52}$$

$$g_n = \langle \psi_0, \phi_{-1,n}\rangle \tag{2.53}$$

Estes resultados levam à fórmula de reconstrução, que gera os coeficientes c_n^j a partir das sequências de decomposição do passo $j+1$.

$$\begin{aligned} c_n^j &= \sum_k h_{n-2k}c_k^{j+1} + \sum_k g_{n-2k}d_k^{j+1} \\ &= \sum_k \left[h_{n-2k}c_k^{j+1} + g_{n-2k}d_k^{j+1}\right] \end{aligned} \tag{2.54}$$

2.4 Tratamento das Condições de Contorno

Na prática as funções (sinais) possuem duração finita. Em razão disto aparece o delicado problema dos efeitos de bordo no instante da aplicação da

convolução com os filtros h ou g nos limites do sinal. Alguns coeficientes destes filtros teriam que ser convoluidos com uma porção do sinal que não existe. As principais maneiras de tratar este problema são:

- a mais simples, completa a sequência que representa a função com zeros (figura 2.13(a));
- possivelmente a mais usada, trata a função como periódica, repetindo o sinal a partir de cada bordo ($x(N+i) \equiv x(i)$)(figura 2.13(b));
- uma outra opção é através da reflexão do sinal nos bordos. Assim, $x(N+i) \equiv x(N \Leftrightarrow i+1)$ e $x(\Leftrightarrow i) \equiv x(i \Leftrightarrow 1)$ (figura 2.13(c));
- finalmente, temos a opção de utilizar funções da base que se adaptem ao intervalo. Em [Gomes and Velho, 1998a] pode ser encontrada uma discussão sobre esta possibilidade.

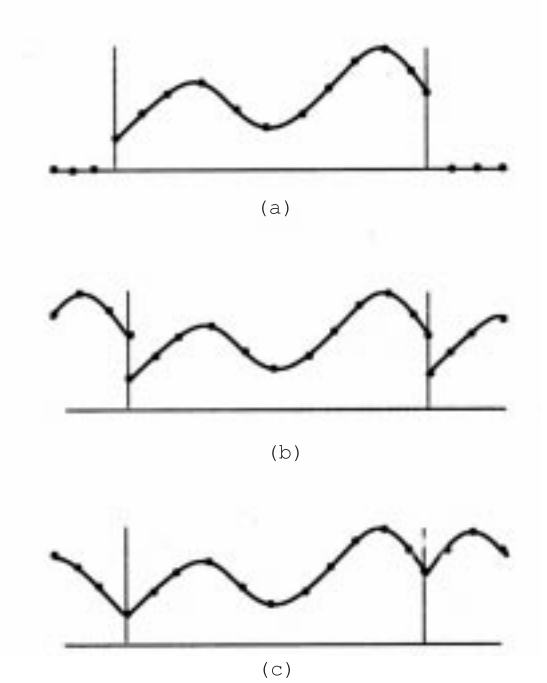

Figura 2.13: Tratamento do Contorno (a) Extensão com Zeros (b) Periodização (c) Reflexão ([Gomes and Velho, 1998a])

2.5 Implementação

2.5.1 Transformada Rápida de *Wavelet*(FWT).

O algoritmo FWT é a implementação direta do esquema da análise de multiresolução, resultando da aplicação direta das equações 2.48. Para compreender a implementação deste algoritmo podemos usar a notação matricial, de modo que estas equações podem ser descritas como:

$$\{W\} = [M]\{f\}, \tag{2.55}$$

onde $\{W\}$ é a transformada de *wavelet* da função $\{f\}$, e $[M]$ é o núcleo da transformação dado por:

$$[M] = \begin{bmatrix} h_0 & \cdots & h_n & \cdots & \cdots & \cdots & \cdots & \cdots \\ g_0 & \cdots & g_n & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & h_0 & h_1 & \cdots & h_n & \cdots & \cdots & \cdots \\ \cdots & g_0 & g_1 & \cdots & g_n & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ h_{i+1} & \cdots & h_n & \cdots & \cdots & h_0 & \cdots & h_i \\ g_{i+1} & \cdots & g_n & \cdots & \cdots & g_0 & \cdots & g_i \end{bmatrix} \tag{2.56}$$

Através da equação 2.55 realizamos um passo da transformação de *wavelet* (um nível de decomposição), obtendo a partir de f_j os coeficientes de *wavelet* $\{c^{j+1}\}$ e $\{d^{j+1}\}$. Podemos então verificar que a sequência $\{W\}$ é dada por:

$$\{W\}^{j+1} = \{c_0^{j+1}, d_0^{j+1}, c_1^{j+1}, d_1^{j+1}, \cdots c_N^{j+1}, d_N^{j+1}\} \tag{2.57}$$

O algoritmo 2.2 apresenta uma implementação da equação 2.55

Algoritmo 2.2

```
ForwardFWT1 (h, g, f, w, nh, ng, nf )
h <=> filtro h da transformada de wavelets com nh elementos
g <=> filtro g da transformada de wavelets com ng elementos
f <=> função a ser transformada com nf elementos
{
    for (i=0; i < nf; i+=2 ) {
        k = i * nh / 2
        w[i] = 0
        for (j=0; j < nh; j++){
            w[i] += h[j] * f[k]
```

```
            k = (k+1)%nf
        }
    }
    for (i=1; i <= nf; i+=2 ){
        k = (i-1) * ng / 2
        w[i] = 0
        for (j=0; j < ng; j++){
            w[i] += g[j] * f[k]
            k = (k+1)%nf
        }
    }
    return w
}
```

Após a execução deste código será obtida uma sequência representando W como a apresentada na equação 2.57, sendo necessário separar as sequências de coeficientes c e d, para gerar a sequência:

$$\{W\}^{j+1} = \{c_0^{j+1}, c_1^{j+1}, \cdots c_N^{j+1}, d_0^{j+1}, d_1^{j+1}, \cdots, d_N^{j+1}\} \tag{2.58}$$

Isto pode ser feito movendo os elementos de $w[i]$ para $w[l]$, com l igual a $l = \frac{nf}{2} + \lfloor \frac{i}{2} \rfloor$. Assim, o código que realiza um nível de decomposição da transformada gerando as sequências $\{c\}$ e $\{d\}$ separadas, é apresentado no algoritmo 2.3.

Algoritmo 2.3

```
ForwardFWT2 (h, g, f, w, nh, ng, nf )
h <=> filtro h da transformada de wavelets com nh elementos
g <=> filtro g da transformada de wavelets com ng elementos
f <=> função a ser transformada com nf elementos
{
    for (i=0; i < nf; i+=2 ){
        k = i * nh / 2
        w[i/2] = 0
        for (j=0; j < nh; j++) {
            w[i/2] += h[j] * f[k]
            k = (k+1)%nf
        }
    }
    for (i=1; i <= nf; i+=2 ) {
        k = (i-1) * ng / 2
        w[i/2 + nf/2] = 0
```

```
        for (j=0; j < ng; j++) {
            w[i/2 + nf/2] += g[j] * f[k]
            k = (k+1)%nf
        }
    }
    return w
}
```

Finalmente, uma outra alteração deve ser realizada para tratar filtros de convolução com origem diferente de 0, ou seja com início das sequências em h_0 e g_0. O algoritmo 2.4 contem estas alterações e gera todos os níveis de decomposição da transformada.

Algoritmo 2.4

```
ForwardFWT (h, g, f, w, nh, ng, h0, g0, L )
h <=> filtro h da transformada de wavelets com nh elementos
h0 <=> origem do filtro h
g <=> filtro g da transformada de wavelets com ng elementos
g0 <=> origem do filtro g
f <=> função a ser transformada com nf elementos
L<=> numero de niveis de decomposicao
{
    nf = 2^L
    for ( l=L-1; l <= 0; l <=> ) {
        for (i=0; i < 2^l; i+=2 ) {
            k = ((i * nh/2) + h0)%2^l
            w[ i/2] = 0
            for (j=0; j < nh; j++) {
                w[ i/2] += h[j] * f[k]
                k = (k + 1)%2^l
            }
        }
        for (i=1; i <= 2^l; i+=2 ) {
            k =((i * ng/2) + g0)%2^l

            w[(i/2) + ng/2] = 0
            for (j=0;j < ng; j++) {
                w[(i/2) + ng/2] += g[j] * f[k]
                k = (k + 1)%2^l
            }
        }
```

```
        copie w para f
    }
    return w
}
```

2.5.2 A FWT Inversa

A transformada inversa recebe uma sequência $\{W\}$ como gerada pelo algoritmo 2.4 e converte em uma representação no espaço de escalas. O algoritmo 2.5 apresenta esta implementação:

Algoritmo 2.5

```
ReverseFWT (h, g, w, nh, ng, h0, g0, nf, L )
h <=> filtro h da transformada de wavelets com nh elementos
h0 <=> origem do filtro h
g <=> filtro g da transformada de wavelets com ng elementos
g0 <=> origem do filtro g
f <=> função a ser transformada com nf elementos
L <=> numero de niveis de decomposicao
{
    tmp = W !! copia os coeficientes para vetor temporario
    for (j=0; j < L; j++) {
        f[0 ··· 2^i(j <=1)] = 0 !! zera o vetor do sinal reconstruido
        for(k=0; k < 2^(j-1); k++) {
            i = ((k <=h0)/2)%2^j
            lb = (k <=h0)%2
            for ( l=lb; l <= nh; l+=2) {
                f[k]+ = h[l] * tmp[i]
                i = CIRC(i <=1, 2^j)
            }
            i = ((k <=g0)/2)%2^j
            lb = (k <=g0)%2
            for ( l=lb; l <= ng; l+=2) {
                f[k]+ = g[l] * tmp[i + 2^j]
                i = CIRC(i <=1, 2^j)
            }
        }
        copia f para tmp
    }
    retorne f
}
```

2.5.3 Representação da Função de Escala e de *Wavelet*

Podemos verificar que para gerar uma transformada discreta de *wavelet* necessitamos apenas do filtro passa baixa h. A partir de h podemos gerar a função $\phi(x)$ associada, o filtro de passa banda g e a função de *wavelet* (*wavelet* básica). Assim, seja h uma sequência que satisfaça:

$$\sum_k h_k = \sqrt{2} \sum_k h_k h_{k+2l} = \delta_l \tag{2.59}$$

Então, existe uma função de escala :

$$\phi(t) = \sum_k h_k \phi(2t \Leftrightarrow k), \tag{2.60}$$

que pode ser construída como uma soma ponderada de cópias em meia escala, usando h como os pesos da ponderação. Ou seja, $\phi(t)$ pode ser calculada numericamente [Daubechies, 1988] através de repetidas convoluções do filtro h com versões escalada da função pulso retangular (2.14), algoritmo cascade. Formalmente temos:

$$\phi(x) = \lim_{x \to \infty} \eta_i(x), \tag{2.61}$$

onde

$$\eta_i(x) = \sqrt{2} \sum_n h(n) \eta_{i-1}(2x \Leftrightarrow n) \tag{2.62}$$

é uma aproximação por partes de $\phi(t)$, e

$$\eta_0(x) = \Pi(x) = \begin{cases} 1, & \mid x \mid < \frac{1}{2} \\ \frac{1}{2}, & \mid x \mid = \frac{1}{2} \\ 0, & \mid x \mid > \frac{1}{2} \end{cases} \tag{2.63}$$

Se iniciarmos o processo com a função ϕ, podemos calcular o filtro h através de:

$$h(k) = \langle \phi_{1,0}(t), \phi_{o,k}(t) \rangle, \tag{2.64}$$

onde

$$\phi_{j,k}(t) = 2^{-\frac{j}{2}} \phi(2^j t \Leftrightarrow k), \text{ com } j = 0, 1, \cdots \text{ e } k = 0, 1, \cdots, 2^j \Leftrightarrow 1 \tag{2.65}$$

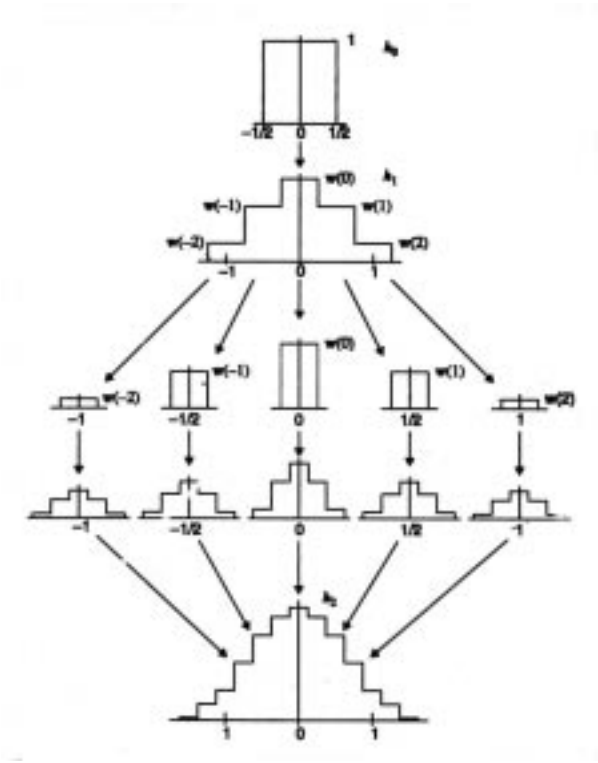

Figura 2.14: Construção da Função de Escala.

O filtro passa banda g pode ser obtido a partir do filtro h usando a expressão

$$g(k) = (\Leftrightarrow 1)^k h(\Leftrightarrow k + 1) \tag{2.66}$$

e, a partir do filtro g obtemos ψ

$$\psi(t) = \sum_k g(k)\phi(2t \Leftrightarrow k) \tag{2.67}$$

A função de *wavelet* ψ também pode ser obtida através da aplicação do algoritmo cascade apresentado para a função de escala.

2.6 Bases de *Wavelets*

Um problema central no desenvolvimento de aplicações baseadas em *wavelet* é a escolha das funções que compõem a base de *wavelets*, as quais dependem

essencialmente das propriedades da *wavelet* mãe. Existe uma grande variedade de bases de *wavelet*, e sua escolha depende dos requisitos da aplicação. Nesta seção apresentaremos as propriedades importantes de bases de *wavelet* e algumas bases mais conhecidas e utilizadas.

2.6.1 Projeto de Bases de *Wavelets*

Como a solução da equação de refinamento possui muitos graus de liberdade, é necessário acrescentar restrições a esta equação para obter a base de *wavelets* com as características(propriedades) necessárias de modo a obter o comportamento desejado para a aplicação. As principais propriedades, que caracterizam o comportamento de uma base de *wavelet* são: ortogonalidade, suporte compacto, simetria, regularidade, número de momentos nulos, interpolação, e coeficientes racionais.

A análise de multiresolução gera funções de *wavelet* e de escala ortogonais ou bi-ortogonais. A característica de ortogonalidade, está associada à capacidade de reconstrução perfeita da transformada, resultando em um processo de decomposição/reconstrução numericamente estável. Nas *wavelet* ortogonais o mesmo par de filtros é usado para decomposição e reconstrução, o que evita a perda de informação, desejável em algumas aplicações. Além disto, os operadores de projeção nos diferentes espaços V_j e W_j resultam em aproximações ótimas das funções de $L^2(\Re)$. A desvantagem das bases ortogonais reside no fato de somente uma delas (Base de Haar) ser formada por funções simétricas. Em função disto podemos em algumas aplicações substituir a base ortogonal por uma base biortogonal, que resulta em uma maior liberdade para o projeto de bases de *wavelet*. Uma outra característica das bases ortogonais é possuir uma ligação direta entre a norma dos coeficientes e a norma da função transformada em L^2, que é dada pela expressão:

$$\|f\| = \sqrt{\sum_{s,t} d_{s,t}^2} \tag{2.68}$$

A transformada rápida de uma base ortogonal é uma transformação unitária, ou seja, a sua adjunta é igual à transposta. Isto é uma importante propriedade em aplicações de análise numérica, pois significa que o erro presente no dado inicial não vai crescer, mantendo o cálculo o mais estável possível.

A largura do suporte das funções ϕ e ψ é determinada pelo comprimento dos coeficientes das sequências (h_n) e (g_n). A característica de suporte compacto é básica para efeito de localização de frequências no sinal. Quanto menor o suporte das funções da base, menor será o número de coeficientes

necssários para a reconstrução de uma parcela do sinal (função). Para *wavelet* ortogonais de suporte compacto, os filtros h e g são filtros de impulso de resposta finito, o que acelera o algoritmo de construção da pirâmide de decomposição. Esta característica influi diretamente no esforço computacional necessário para o cálculo da transformada, pois se o suporte for muito largo é grande o número de amostras envolvidas na convolução dos filtros com o sinal.

O número de momentos nulos, definido como o maior inteiro n que anula as integrais:

$$\langle x^n \rangle_\psi = \int_{-\infty}^{\infty} \psi(x) x^n \, dx, \; n = 0 \ldots p \Leftrightarrow 1, \tag{2.69}$$

determina a taxa de convergência da aproximação por *wavelet* de funções regulares. Esta propriedade também caracteriza a regularidade da *wavelet*, sendo bastante importante na detecção de singularidades. Este número determina a razão de convergência de aproximações por *wavelet* de funções regulares.

A característica de regularidade das funções da base é desejada em aplicações que envolvem o cálculo de derivadas e compressão. Esta característica está associada a uma melhor localização das frequências dos filtros usados, o que resulta em melhores taxas de compressão. Por outro lado, a remoção de coeficientes abaixo de um certo limite será menos percebida se forem usadas *wavelet* regulares. Em bases não ortogonais podemos verificar que esta característica é mais relevante para a base primária que para a dual.

A simetria é importante em aplicações como processamento de imagens e visão computacional. Esta propriedade pode ser explorada na quantização de imagens para compressão e no processamento dos bordos de sinais de duração limitada usando periodização por reflexão do dado. Segundo demonstrado em [Daubechies, 1992], simetria e ortogonalidade são características incompatíveis, exceto para a *wavelet* de Haar. Como decorrência da simetria da função da base, verifica-se que os filtros são simétricos, denominados filtros de fase linear. Isto garante que não haverão distorções de fase durante a reconstrução.

Quando os coeficientes dos filtros são números racionais, a implementação computacional se torna mais eficiente e precisa. Melhor ainda se forem números racionais diádicos, assim podemos substituir a divisão por dois, por um *shift* de bits.

Geralmente, a função analítica das funções de escala e de *wavelet* não estão disponíveis. Mas em certos casos elas podem ser bastante úteis. Um importante exemplo de base de *wavelet* com esta característica é a formada pelas B-splines. A propriedade de interpolação torna possível o cálculo de

coeficientes de escala em qualquer posição do domínio, a partir de amostras da função. Quando a função de escala $\phi(k) = \delta_k$, para $k \in Z$, os coeficientes $c_k^j = \langle f, \phi_j \rangle$ da projeção P_{V_j} em V_j, são os valores da função $f(2^j k)$, amostrados em um reticulado com espaçamento 2^{-j}

Como estas diferentes propriedades não são inerentes a uma única base de *wavelet*, algumas são conflitantes, surgindo então diferentes bases apropriadas para cada aplicação. Assim, vamos apresentar a seguir as classes de bases de *wavelet* mais comuns e suas principais características.

2.6.2 Bases Ortogonais

Na maioria das aplicações, as bases ortogonais representam a melhor escolha, embora sejam não simétricas (exceto a de Haar). A simetria das funções da base pode ser reduzida no projeto da base de funções. Estas bases resultam como consequência direta da análise de multiresolução, onde os espaços W_j são definidos como complemento ortogonal de V_j em V_{j-1}. A condição que garante a ortogonalidade pode ser expressa por:

$$\langle \psi, \phi(\Leftrightarrow l) \rangle = 0, \tag{2.70}$$

ou seja

$$W_0 \perp V_0 \tag{2.71}$$

Entre as bases ortogonais mais comuns podemos citar as bases de Haar, Daubechies, e Coiflet.

Base de Haar

A *wavelet* mais simples é a de Haar. As funções desta base são definidas por:

$$\psi(x) = \begin{cases} 1, & \text{se } x \in \left[0, \frac{1}{2}\right] \\ \Leftrightarrow 1, & \text{se } x \in \left[\frac{1}{2}, 1\right] \\ 0, & \text{se } x < 0 \text{ ou } x > 1 \end{cases} \tag{2.72}$$

$$\psi_{s,t}(x) = 2^{-\frac{s}{2}} \psi(2^{-s} u \Leftrightarrow t), m, n \in Z \tag{2.73}$$

A função de escala associada, pode ser representada por:

$$\phi(x) = \begin{cases} 0, \text{ se } x < 0 \text{ ou } x > 1 \\ 1, \text{ se } x \in [0, 1] \end{cases} \tag{2.74}$$

	0	1
h	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$
g	$\frac{1}{\sqrt{2}}$	$\Leftrightarrow\frac{1}{\sqrt{2}}$

Tabela 2.1: Filtros da Base de Haar

Podemos verificar que cada função $\psi_{s,t}(x)$ tem seu suporte nos intervalos fechados $I_{s,t} = [2^{-m}n, 2^{-m}]$. Os filtros que definem esta transformada estão descritos na tabela 2.1.

As funções de Haar não são contínuas, nem regulares, mas representam as *wavelet* mais compactas, possuindo boa localização no espaço. Adicionalmente, possuem a vantagem de não precisar de processamento numérico para determinar valores arbitrários das funções da base.

Base de Daubechies

Em [Daubechies, 1988], é apresentada uma família de *wavelet*, $\{D_k^r(x)\}$, que é ortonormal e possui suporte compacto. Para cada valor do inteiro r, o conjunto de *wavelets*

$$\{D_{j,k}^r(x)\} = \{2^{\frac{j}{2}} D^r(2^j x \Leftrightarrow k)\}, \text{ com } j,k \in Z, \tag{2.75}$$

forma uma base de *wavelets* ortonormal. Podemos verificar que $D^r(x)$ é zero fora do intervalo $[0, 2r \Leftrightarrow 1]$, e os r primeiros momentos se anulam, ou seja:

$$\int_{-\infty}^{\infty} x^n D^r(x)\, dx = 0, \;\; n = 0, 1, \cdots, r, \tag{2.76}$$

também temos que o número de derivadas contínuas é aproximadamente $\frac{r}{5}$.

É interessante notar que D^1 é a *wavelet* básica de Haar, e a regularidade aumenta com o número de momentos nulos. As *wavelet* de Daubechies possuem o menor suporte para um dado número de momentos nulos. Para uma descrição dos valores dos filtros de várias *wavelets* de Daubechies ver [Sévenier, 1994]. A figura 2.15 mostra as funções de escala e *wavelets* D^2 a D^7.

2.6.3 Bases Biortogonais

A condição de ortogonalidade impõem uma forte restrição para a construção de *wavelets*. Por exemplo, sabe-se que a *wavelet* de Haar é a única *wavelet* ortogonal, simétrica e com suporte compacto. A generalização para *wavelets* biortogonais aumenta a flexibilidade no projeto de bases de *wavelets*.

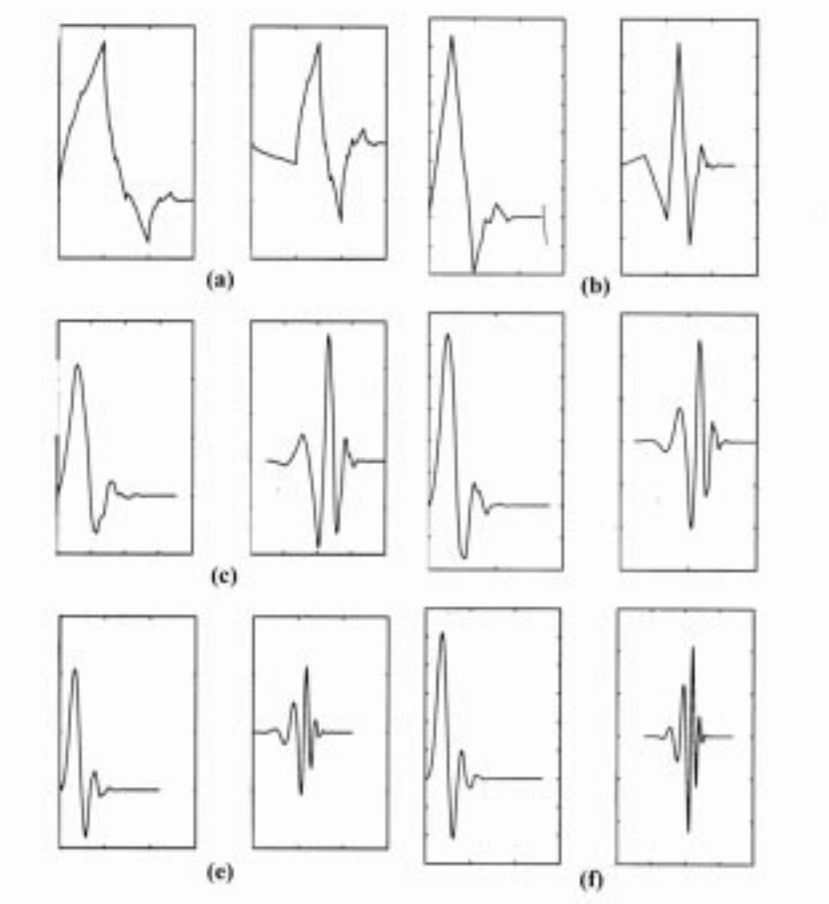

Figura 2.15: Funções de escala e de *wavelets* da família de Daubechies

A análise de multiresolução biortogonal emprega operadores de projeção similares aos apresentados nas equações 2.40, sendo expressos por:

$$\begin{aligned} P_{V_j} f &= \sum_n \langle f, \phi_{jn} \rangle \tilde{\phi}_{jn} \\ P_{W_j} f &= \sum_n \langle f, \psi_{jn} \rangle \tilde{\psi}_{jn}, \end{aligned} \tag{2.77}$$

onde os pares de funções ϕ, ψ, e $\tilde{\phi}$, $\tilde{\psi}$ são usados como funções para decomposição e reconstrução respectivamente, sendo denominadas pares de funções primal e dual.

As funções duais de *wavelet* $\tilde{\psi}$ e de escala $\tilde{\phi}$, geram uma análise dual de

multiresolução com subespaços $\tilde{V}_j$ e $\tilde{W}_j$, tal que:

$$\tilde{V}_j \perp W_j \text{ e } V_j \perp \tilde{W}_j, \tag{2.78}$$

e consequentemente:

$$\tilde{W}_j \perp W_{j'} \text{ para } j \neq j' \tag{2.79}$$

A análise de multiresolução dual não necessáriamente é a mesma gerada pela base de funções original. De uma maneira mais geral, podemos afirmar que:

$$\begin{aligned} \langle \tilde{\phi}_{jl}, \phi_{jl'} \rangle &= \delta_{l-l'},\ l', l \in Z \\ \langle \tilde{\psi}_{jl}, \psi_{j'l'} \rangle &= \delta_{j-j'}, \delta_{l-l'},\ l', l, j, j' \in Z \end{aligned} \tag{2.80}$$

As definições para $\tilde{\psi}$ e $\tilde{\phi}$ são as mesmas usadas para ϕ e ψ, ou seja, as funções devem satisfazer:

$$\begin{aligned} \tilde{\phi}(x) &= 2 \sum_k \tilde{h}_k \tilde{\phi}(2x \Leftrightarrow k) \\ \tilde{\psi}(x) &= 2 \sum_k \tilde{g}_k \tilde{\phi}(2x \Leftrightarrow k) \end{aligned} \tag{2.81}$$

Assim, das equações 2.80 e 2.81 temos que:

$$\begin{aligned} \tilde{h}_{k-2l} &= \langle \tilde{\phi}(x \Leftrightarrow l), \phi(2x \Leftrightarrow k) \rangle \\ \tilde{g}_{k-2l} &= \langle \tilde{\psi}(x \Leftrightarrow l), \phi(2x \Leftrightarrow k) \rangle \end{aligned} \tag{2.82}$$

Um par de funções (escala e *wavelet*) biortogonal é denominado semiortogonal se geram uma análise de multiresolução ortogonal [Chui, 1992]. A denominação pré-*wavelet* também é usada. Posto que os espaços W_j são mutualmente ortogonais, temos que:

$$W_j \perp \tilde{W}_{j'} \text{ e } W_j \perp W_{j'} \text{ para } j \neq j' \tag{2.83}$$

Consequentemente, $W_j = W_{j'}$, o que implica que $V_j = V_{j'}$. Assim, as funções primal e dual geram a mesma análise de multiresolução (ortogonal). Dentre as vantagens das bases de *wavelets* biortogonais, podemos destacar a capacidade de projetar uma base de funções simétricas.

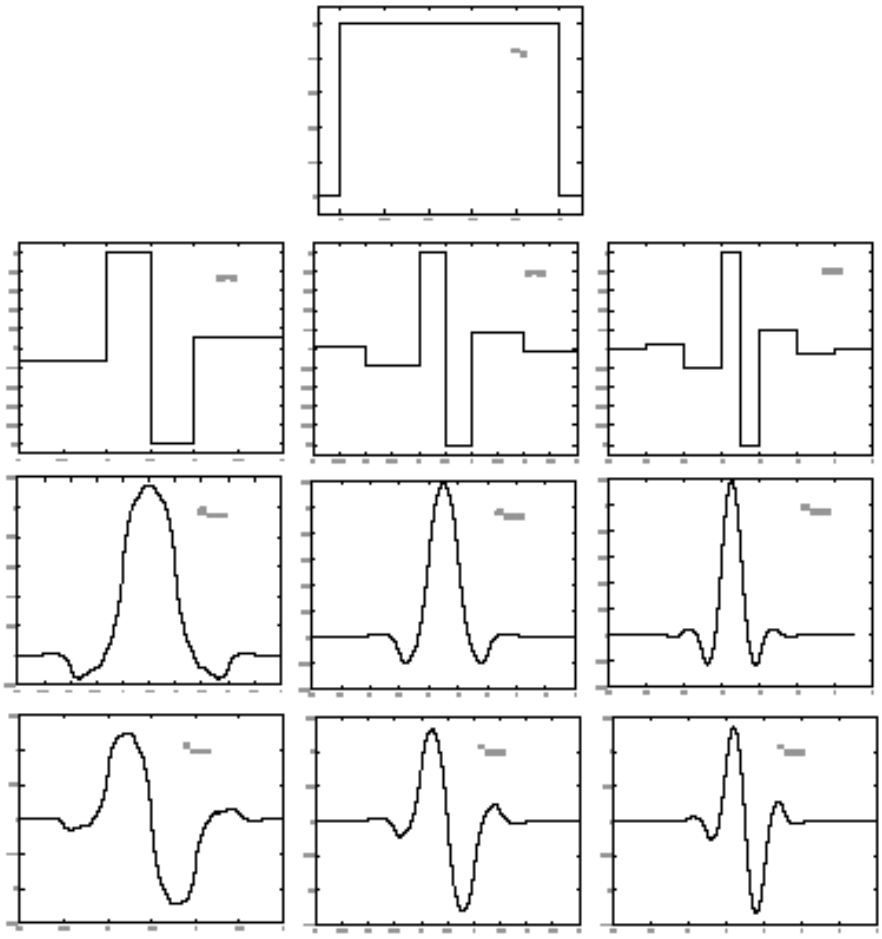

Figura 2.16: Funções de escala e de *wavelets* biortogonais.

Base de Splines

A base de splines, proposta por [Chui and Wang, 1991], é uma base biortogonal que possui várias propriedades interessantes como: suavidade e momentos nulos. Além disto, estão disponíveis as expressões analíticas para a função de escala e de *wavelet* e suas duais, tanto no domínio espacial quanto no da frequência. A figura 2.16 apresenta algumas funções de escala e de *wavelets* Bsplines.

A função de escala de ordem 0 é dada por:

$$\phi^0(x) = \begin{cases} 1, \text{ se } 0 \leq x < 1 \\ 0, \text{ de outra forma} \end{cases} \tag{2.84}$$

Funções de escala com suporte compacto de ordem j podem ser geradas recursivamente em termos de B-splines cardinais:

$$\phi^j(x) = \frac{x}{j \Leftrightarrow 1}\phi^{j-1}(x) + \frac{j \Leftrightarrow x}{j \Leftrightarrow 1}\phi^{j-1}(x \Leftrightarrow 1) \tag{2.85}$$

No domínio da frequência as funções correspondentes são descritas por

$$\hat{\phi}^j(f) = \left(\frac{1 \Leftrightarrow e^{-i2\pi f}}{i2\pi f}\right)^j \tag{2.86}$$

A função de *wavelet* é expressa por:

$$\psi^j(f) = \sum_{k=0}^{3j-2} q_{j,k}\phi^j(2x \Leftrightarrow k), \tag{2.87}$$

com

$$q_{j,k} = \frac{(\Leftrightarrow 1)^k}{2j \Leftrightarrow 1}\sum_{l=0}^{j}\left(\frac{j}{l}\phi^{2j}(K+1 \Leftrightarrow l)\right) \tag{2.88}$$

Pode ser verificado que o $supp(\psi^j) = [0, 2j \Leftrightarrow 1]$. Definições para as funções duais podem ser encontradas em [Chui, 1992]

2.7 Transformada de *Wavelets* 2D e 3D

As transformadas de *wavelets* 2D e 3D podem ser obtidas por considerações de separabilidade, o que pode ser feito através da aplicação de um certo número de transformadas unidimensionais. Por exemplo, para uma transformada 2D podemos aplicar a transformada às linhas do sinal 2D, e em seguida às colunas. Assim, a teoria de análise de multiresolução pode ser facilmente extendida para espaços 2D e 3D. Para uma aproximação em multiresolução n-dimensional, cada espaço de aproximação $V_j^n \in L^2(\Re)$ pode ser decomposto como um produto tensorial de n espaços unidimensionais idênticos $V_j \in L^2(\Re)$ por:

$$V_j^n = V_j \otimes \cdots \otimes V_j \tag{2.89}$$

2.7.1 Transformada 2D

Para n igual a dois, podemos extender a análise de multiresolução através do seguinte teorema:

Se ϕ é a base de V_j, então V_j^2 é construído através do produto tensorial $\phi(x,y) = \phi(x)\phi(y)$. Se $\phi(x,y)$ é a função de escala bidimensional de uma aproximação em multiresolução de $L^2(\Re)$, e $\psi(x)$ é a *wavelet* unidimensional associada com a função $\phi(x)$, então os espaços de diferenças bidimensionais W_j^2 são construídos a partir de $3*(2^n \Leftrightarrow 1)$ *wavelet* bidimensionais, dadas por:

$$\begin{aligned} N_{l,m}^{0,j}(x,y) &= \psi_l^j(x)\psi_m^j(y) \\ N_{l,m}^{1,j}(x,y) &= \psi_l^j(x)\phi_m^j(y) \\ N_{l,m}^{2,j}(x,y) &= \phi_l^j(x)\psi_m^j(y), \end{aligned} \tag{2.90}$$

e

$$L^2(\Re^2) = \bigoplus_{j=-\infty}^{\infty} W_j^2. \tag{2.91}$$

Se considerarmos uma imagem $n \times n$, $f_j(x,y)$, onde n é uma potência de 2, e j representa a escala da função. Podemos representá-la em termos das *wavelet* bidimensionais. Em cada estágio da transformação decompomos a imagem em quatro subimagens de resolução $n/2 \times n/2$ como mostrado na figura 2.17. Cada uma destas imagens é gerada pelo produto vetorial da imagem com uma das funções da base 2D de *wavelet*, seguida de uma subamostragem em x e y. Para o primeiro passo podemos escrever:

$$\begin{aligned} f_1^0 &= \langle f_0^0(x,y), \phi(x \Leftrightarrow 2m, y \Leftrightarrow 2n)\rangle \\ f_1^1 &= \langle f_0^0(x,y), N_{2,j}(x \Leftrightarrow 2m, y \Leftrightarrow 2n)\rangle \\ f_1^2 &= \langle f_0^0(x,y), N_{1,j}(x \Leftrightarrow 2m, y \Leftrightarrow 2n)\rangle \\ f_1^3 &= \langle f_0^0(x,y), N_{0,j}(x \Leftrightarrow 2m, y \Leftrightarrow 2n)\rangle \end{aligned} \tag{2.92}$$

Para passos subsequentes, $f_{2^j}^0(x,y)$ é decomposta da mesma forma para gerar imagens na escala 2^{j+1}.

Escrevendo os produtos internos como convoluções temos:

$$\begin{aligned} f_1^0 &= \left\{\left[f_{2^j}^0(x,y) * \phi(\Leftrightarrow x, \Leftrightarrow y)\right](2m, 2n)\right\} \\ f_1^1 &= \left\{\left[f_{2^j}^0(x,y), N_{2,j}(\Leftrightarrow x, \Leftrightarrow y)\right](2m, 2n)\right\} \\ f_1^2 &= \left\{\left[f_{2^j}^0(x,y), N_{1,j}(\Leftrightarrow x, \Leftrightarrow y)\right](2m, 2n)\right\} \\ f_1^3 &= \left\{\left[f_{2^j}^0(x,y), N_{0,j}(\Leftrightarrow x, \Leftrightarrow y)\right](2m, 2n)\right\} \end{aligned} \tag{2.93}$$

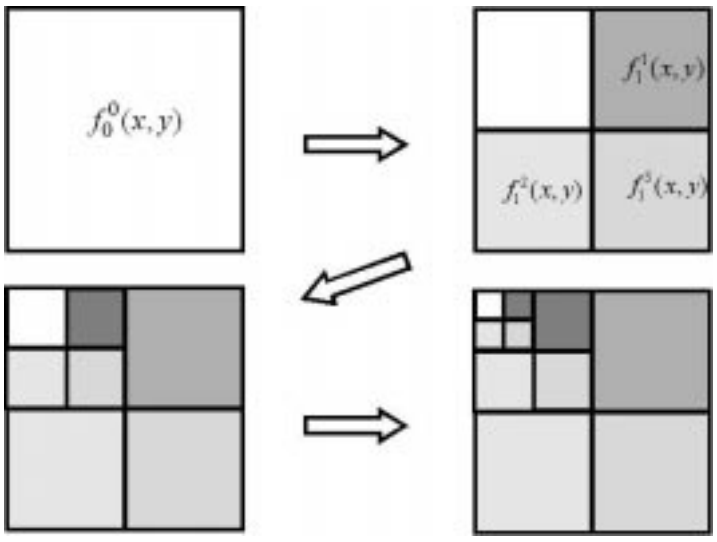

Figura 2.17: Transformada de *Wavelets* Discreta 2D

Como as funções de escala e de *wavelet* são separáveis, cada convolução bidimensional pode ser quebrada em uma convolução ao longo das linhas e outra ao longo das colunas de f, conforme mostrado na figura 2.18.

No primeiro estágio realizamos uma convolução das linhas da imagem $f_{2^j}^0$ com o filtro h e com o filtro g, colocando os coeficientes c e d em suas posições respectivas, de modo a gerar as sequências apresentadas na equação 2.58 para sinais unidimensionais. Em seguida repetimos o processo para as colunas, gerando assim as quatro subimagens. Se considerarmos que os filtros h e g são filtros ideais passa banda e passa baixa, temos que $f_0^{2^j}$ contém a informação de baixa frequência da escala anterior, $f_1^{2^j}$,$f_2^{2^j}$, e $f_3^{2^j}$ contém a informação acerca das arestas horizontais, verticais e diagonais respectivamente, estes filtros são denominados anisotrópicos.

A transformada inversa é realizada por um processo similar. Em cada passo do algoritmo combinamos as sequências de coeficientes das linhas das imagens dos níveis subjacentes e em seguida realizamos o mesmo para as colunas (figura 2.19).

2.7.2 Transformada 3D

Podemos extender a análise de multiresolução para três dimensões. Assim, se ϕ é a função da base de V_j, então V_j^3 é construído através do produto tensorial $\phi(x,y,z) = \phi(x)\phi(y)\phi(z)$. Se $\phi(x,y,z)$ é a função de escala tridimensional de uma aproximação em multiresolução de $L^2(\Re)$, e $\psi(x)$ é a *wavelet* uni-

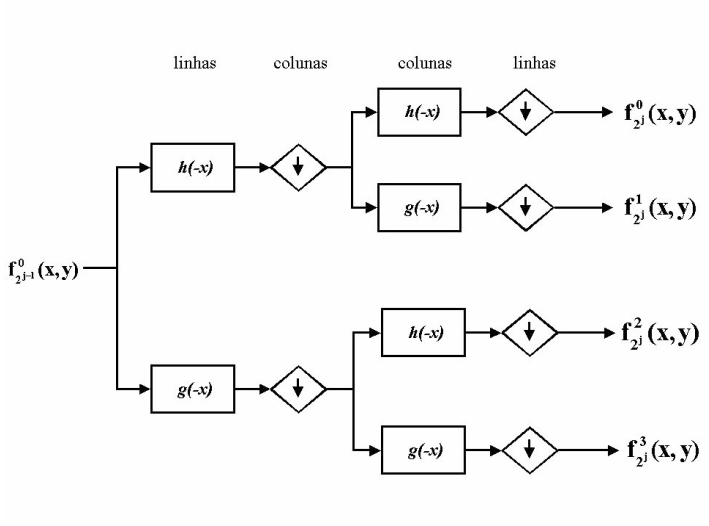

Figura 2.18: Passo da transformada de *Wavelets* Discreta 2D

dimensional associada com a função $\phi(x)$, então os espaços de diferenças tridimensionais W_j^3 são construídos a partir de sete *wavelet* tridimensionais, dadas por:

$$\begin{aligned}
N^{0,j}_{l,m,n}(x,y,z) &= \psi^j_l(x)\psi^j_m(y)\psi^j_n(z) \\
N^{1,j}_{l,m,n}(x,y,z) &= \psi^j_l(x)\psi^j_m(y)\phi^j_n(z) \\
N^{2,j}_{l,m,n}(x,y,z) &= \psi^j_l(x)\phi^j_m(y)\psi^j_n(z) \\
N^{3,j}_{l,m,n}(x,y,z) &= \psi^j_l(x)\phi^j_m(y)\phi^j_n(z) \\
N^{4,j}_{l,m,n}(x,y,z) &= \phi^j_l(x)\psi^j_m(y)\psi^j_n(z) \\
N^{5,j}_{l,m,n}(x,y,z) &= \phi^j_l(x)\psi^j_m(y)\phi^j_n(z) \\
N^{6,j}_{l,m,n}(x,y,z) &= \phi^j_l(x)\phi^j_m(y)\psi^j_n(z),
\end{aligned} \tag{2.94}$$

e

$$L^2(\Re^3) = \bigoplus_{j=-\infty}^{\infty} W_j^3. \tag{2.95}$$

O algoritmo tridimensional para decomposição de um volume segundo uma base de *wavelet* é obtido através da aplicação dos filtros h e g ao longo dos eixos z, y e x repetidamente, como esquematizado na figura 2.20

O resultado deste processo são 7 subvolumes de detalhes $D^{d1}_{mf}f \cdots D^{d7}_{mf}f$ e um sinal resultante de uma filtragem por um passa baixa $A_m f$ que será

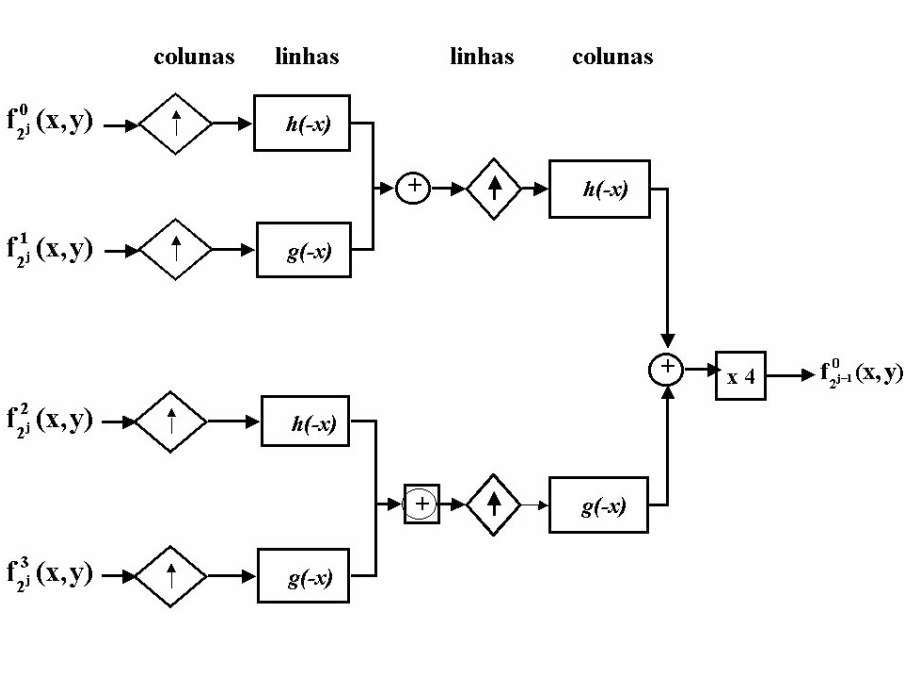

Figura 2.19: Transformada Inversa 2D.

decomposto nos passos posteriores. A pirâmide resultante da aplicação deste processo pode ser vista na figura 2.21.

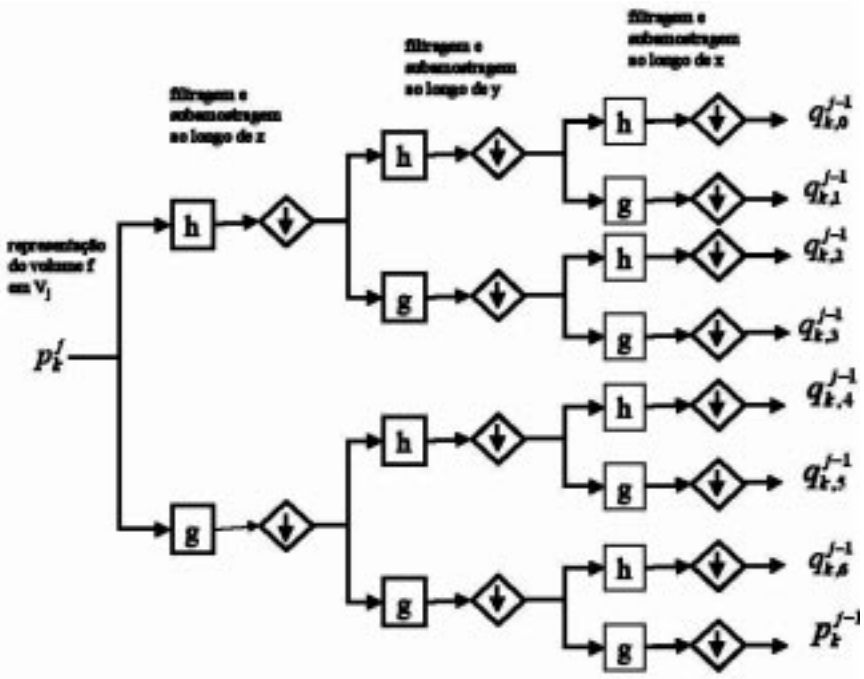

Figura 2.20: Esquema de Convolução para *Wavelet* 3D

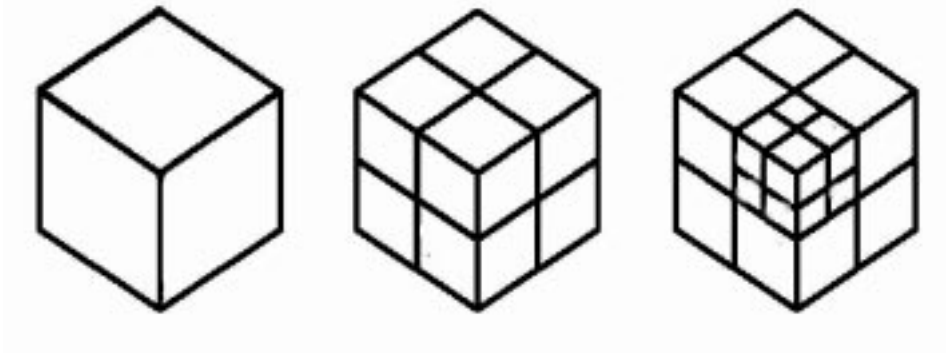

Figura 2.21: Decomposição em *Wavelets* do Volume

Capítulo 3

Visualização Volumétrica

3.1 Introdução

Podemos conceituar visualização volumétrica como o conjunto de técnicas de visualização relacionadas com a representação, manipulação e visualização de conjuntos de dados, denominados dados volumétricos. Estes dados podem ser enquadrados dentro do conceito de objeto gráfico [Gomes and Velho,], sendo assim denominados objetos volumétricos. Um objeto volumétrico (objeto gráfico 3D) pode ser definido como uma função $f : U \subset \Re^3 \leftrightarrow \Re^m$, onde U é o suporte geométrico do objeto e f é a função de atributos função de densidade). Assim, verificamos que é a função de densidade que descreve o objeto volumétrico.

Baseados nos níveis de abstração do paradigma dos quatro universos proposto em [Gomes and Velho,], podemos verificar que o processamento de objetos gráficos é baseado na descrição, representação e reconstrução do objeto, e nas estruturas de dados utilizadas para sua manipulação.

Neste trabalho vamos considerar como dados volumétricos os objetos gráficos volumétricos representados no esquema por matriz [Gomes and Velho, 1998b]. Neste esquema a função de densidade é representada por uma matriz onde cada célula (*voxel*) contém a informação de densidade e os outros atributos do objeto. Assim, podemos dizer que os dados volumétricos aqui tratados, são dados escalares ou vetoriais amostrados em uma grade tridimensional regular

Estes conjuntos de dados tem sido largamente utilizados em muitas aplicações científicas e de engenharia, tais como dinâmica dos fluidos, ciências dos materiais e em várias técnicas de diagnósticos de patologias por imagens. Também a área de síntese de imagens realistas tem utilizado as técnicas de visualização volumétrica para gerar imagens de fenômenos amorfos, tais como

chamas, nuvens e gases.

Dentre as técnicas de visualização volumétrica existem duas classes básicas: extração de superfícies (*surface fitting*) e visualização direta de volumes. Estas duas abordagens diferem basicamente pela utilização ou não de representações intermediárias dos dados volumétricos para a geração da visualização adequada. Nos métodos de extração de superfície é gerada uma representação por polígonos dos dados volumétricos, a partir da qual é realizada a visualização com os métodos de visualização de polígonos. Os métodos de visualização direta de volumes não representam os objetos 3D como superfícies geométricas e arestas. Os dados volumétricos são visualizados diretamente, sendo tratados como um objeto amôrfo semitransparente.

A seguir descreveremos as técnicas básicas para visualização direta de volumes, e os dois principais algoritmos representantes destas técnicas: *ray casting* e *splatting*.

3.2 Visualização Direta de Volumes

Na visualização direta de volumes, a imagem é gerada através de um processo de composição dos valores dos *voxels* ao longo da direção de visualização. Este princípio básico está ilustrado na figura 3.1, que representa como o valor de um *pixel* é obtido através da composição dos *voxels* interceptados pelo raio de visão associado.

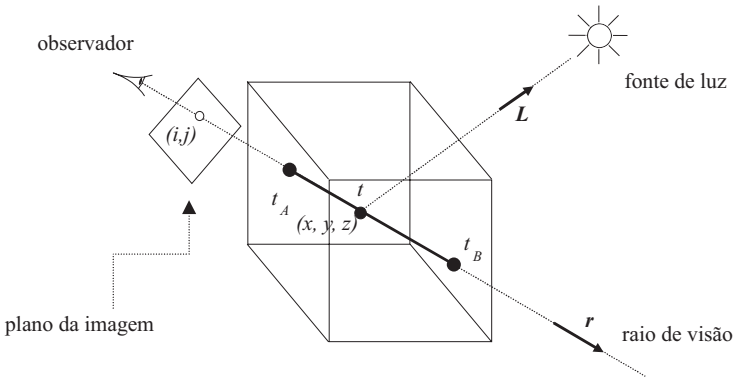

Figura 3.1: Principio básico de rendering de volumes

O processo de visualização de volumes pode ser definido como a criação de uma imagem $p(x_i, y_j)$ que representa a projeção da função $f(x_i, y_j, z_k)$(função de densidade do objeto volumétrico) segundo uma direção especificada. Este

processo, em geral, é realizado através de três etapas básicas (diagrama da figura 3.2):

- Classificação
- Cálculo da Iluminação
- Formação da Imagem

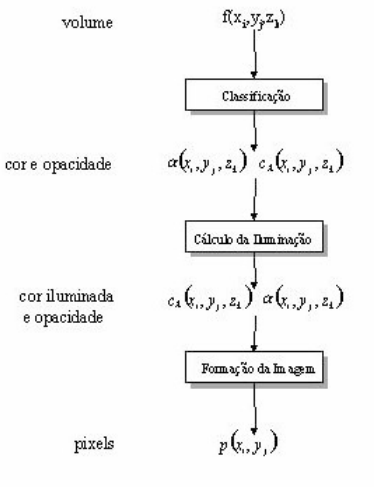

Figura 3.2: Etapas básicas da visualização direta de volumes

A primeira etapa é a classificação dos dados. Esta etapa permite ao usuário encontrar as estruturas existentes no conjunto de dados, permitindo isolá-las, definir sua forma e extensão. A classificação envolve o mapeamento dos valores escalares a serem visualizados, em valores de cor e opacidade associados. Isto é feito através de funções de mapeamento que são específicas para cada aplicação (funções de transferência).

A opacidade é uma medida da transparência de cada *voxel*, descrevendo a quantidade de luz incidente no *voxel* que é absorvida por ele. Em geral esta propriedade varia entre 0 e 1, sendo o valor 1 associado a *voxels* que absorvem toda a luz incidente (completamente opacos).

O mapeamento do valor escalar em cor, é feito para gerar uma representação visual do material que representa cada *voxel*. Já o mapeamento em valores de opacidade, é usado para possibilitar a exibição ao observador apenas da parte do volume que lhe interessa, tornando transparentes as porções

que não precisarem ser visualizadas, e permitindo a visualização de estruturas internas ao volume. Resumindo, a função do processo de classificação é possibilitar a identificação do material que compõem cada *voxel*.

O cálculo da iluminação é a etapa do processo de visualização que tem o objetivo de descrever a aparência dos objetos. Este processo envolve a descrição da interação dos diferentes tipos de luz, de diferentes fontes luminosas, em diferentes posições, com os materiais dos *voxels* que compõem o volume. Esta etapa modifica a cor especificada para cada *voxel* pelo mapeamento da função de transferência, de modo a realçar a percepção tridimensional na imagem final. Em geral são usadas adaptações dos modelos de iluminação tradicionais da computação gráfica (e.g. Phong).

A terceira etapa envolve o processo de formação da imagem com a projeção de um volume de valores escalares representando a cor de cada *voxel*, já considerando a iluminação. Nesta etapa verifica-se como cada *voxel* contribui para o valor dos *pixels* . Este processo envolve a utilização dos métodos de reconstrução volumétrica (interpolação) e uma modelagem da forma como a imagem do volume será formada.

Nas seções seguintes detalharemos cada uma das etapas deste processo de visualização, usando para isto uma abordagem similar à utilizada pelo algoritmo *ray casting*.

3.2.1 Classificação

O passo de classificação é a etapa inicial do processo de visualização volumétrica. Isto se dá por que a manipulação do volume de dados é independente da técnica que vai ser utilizada para visualização, sendo utilizada como uma etapa de pré-processamento para adequação do volume de dados. Em algoritmos de visualização direta de volumes, a classificação é feita através de um mapeamento dos valores dos *voxels* em valores de cor e opacidade. Esta é uma tarefa sujeita a erros, e exige que o usuário possua um certo conhecimento acerca do que espera encontrar no volume. Em aplicações como imagens médicas, os dados e valores de componentes são razoavelmente conhecidos, assim este não é um problema crítico. No entanto, para aplicações como interpretação sísmica, esta etapa é crítica, pois a priori o analista não sabe o que irá encontrar no volume. Uma ferramenta eficiente para o auxilio do projeto da função de mapeamento é o histograma do volume. O histograma representa o número de vezes que ocorrem *voxels* de cada um dos valores possíveis (valores discretos de *voxels*). Assim pode-se conhecer um pouco da estrutura dos dados do volume e decidir os valores que representam a estrutura que se deseja visualizar.

3.2.2 Cálculo da Iluminação

O processo de iluminação é utilizado para melhorar a aparência 3D da imagem através da criação de uma ilusão de profundidade. Esta ilusão de profundidade é obtida através da variação da cor dos objetos de acordo com: sua distância em relação ao plano da imagem; material constituinte; e orientação com respeito a fonte de luz e ao observador. A forma de efetuar estes cálculos da cor dos objetos é definida de acordo com um modelo, denominado modelo de iluminação.

Os modelos de iluminação para computação gráfica são divididos em duas classes: locais e globais. Os modelos globais geram imagens de melhor qualidade, pagando o preço de serem computacionalmente mais caros. Em geral estes modelos tentam levar em consideração a luz refratada que é transmitida através dos objetos, e as componentes especular e difusa da luz refletida. Alguns trabalhos propõem a utilização de modelos globais de iluminação para a visualização de volumes em aplicações como sistemas de visualização científica e cenas sintetizadas. No entanto a maioria dos sistemas de visualização volumétrica utiliza os métodos de iluminação local. Os modelos locais de iluminação baseiam-se apenas na reflexão da luz na superfície do objeto. A figura 3.3 apresenta as componentes de iluminação na interação da luz com a superfície do objeto. Nesta figura fica claro que é fundamental a determinação da orientação da superfície do objeto a ser visualizado. Estes modelos somente computam a reflexão difusa e especular da luz emitida pelas fontes luminosas sobre a superfície do objeto.

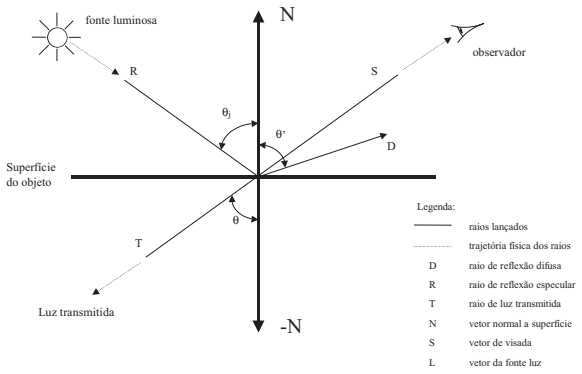

Figura 3.3: Relações da luz com a superfície do objeto.

Um modelo bastante popular foi proposto por Phong [Phong, 1975]. Neste modelo a reflexão difusa é causada pela absorção e reirradiação uniformemente distribuída da luz a partir da superfície iluminada. Este efeito é

modelado pela lei de Lambert, descrita pela equação:

$$I = I_I k_d (N \cdot L). \tag{3.1}$$

Nesta equação I é a intensidade refletida, I_I é a luz incidente, k_d é o coeficiente de reflexão difusa característico do material, e N é o vetor normal à superfície no ponto em questão. Para situações práticas, a fonte de luz é modelada como um termo constante (luz ambiente) e um termo linear, assim o modelo pode ser descrito por:

$$I = I_a k_a + I_I k_d (N\dot{L}), \tag{3.2}$$

onde I_a é a intensidade da fonte de luz e k_a é a constante de reflexão ambiental. Se considerarmos ainda a componente especular, temos o modelo de iluminação expresso pela equação:

$$I = I_a k_a + \frac{I_I (k_d (N \cdot L) + k_s (E \cdot R)^n)}{D + K}, \tag{3.3}$$

onde d é a distância entre o observador e o objeto, K é uma constante arbitrária, R é o vetor da fonte de luz, E é o vetor do observador, k_s é o coeficiente de reflexão especular, e n é o expoente de reflexão especular.

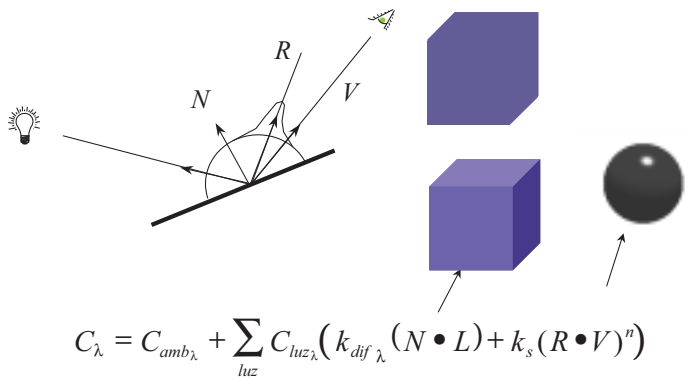

Figura 3.4: Modelo de iluminacao de Phong

Definido o modelo de iluminação podemos utilizar um dos dois algoritmos de iluminação: Gourad [Gouraud, 1971] ou [Phong, 1975]. O algoritmo de Gourad fornece a intensidade de luz no centro de cada *voxel* do volume, então usa este valor para interpolar a iluminação em pontos intermediários. O algoritmo de Phong, interpola o valor da normal e calcula a iluminação em cada ponto intermediário que surja. Estes modelos são utilizados para calcular a cor de cada *voxel* do volume. Isto está representado na figura 3.4.

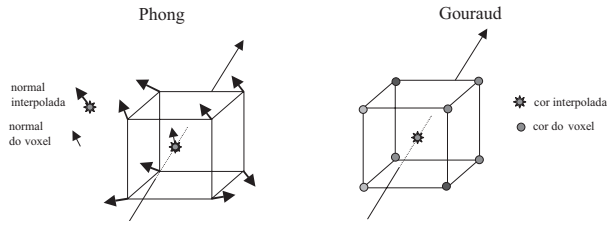

Figura 3.5: Algoritmos de iluminacao de Gouraud e Phong

Um ponto central à aplicação dos algoritmos de iluminação é a necessidade de utilização da orientação da superfície. Isto é feito através da estimativa do vetor normal à superficie no ponto em que será iluminado. Quando se trata de visualização de volumes não faz muito sentido falar em superfícies. No entanto considera-se no cálculo da iluminação que o ponto a ser iluminado pertence a uma isosuperfície existente no volume e que passa pelo *voxel*. A figura 3.6 apresenta uma representação desta questão.

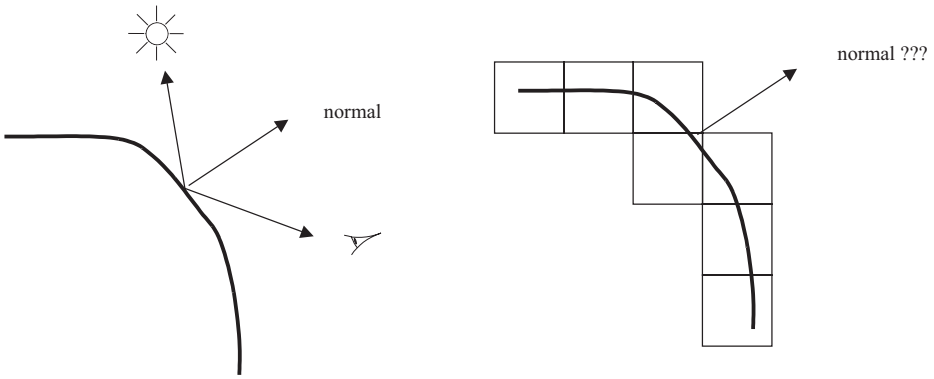

Figura 3.6: Estimativa da normal para iluminação

Considere o dado volumétrico, representado por V, um campo escalar diferenciável. Então para V definido numa grade regular, o gradiente pode ser aproximado através de diferenças finitas por:

$$
\begin{aligned}
V &= (V_x, V_y, V_z), \text{ onde} \\
V_x &= \frac{1}{2}\left(V\left(x_{i+1}, y, z\right) \Leftrightarrow V\left(x_{i-1}, y, z\right)\right), \\
V_y &= \frac{1}{2}\left(V\left(x, y_{j+1}, z\right) \Leftrightarrow V\left(x, y_{j-1}, z\right)\right), \\
V_z &= \frac{1}{2}\left(V\left(x, y, z_{k+1}\right) \Leftrightarrow V\left(x, y, z_{k-1}\right)\right)
\end{aligned} \tag{3.4}
$$

Desta forma a determinação da normal N é feita baseada no valor do campo escalar que representa o volume, e as componentes desta normal são:

$$
\begin{aligned}
N_x &= V(x+1, y, z) \Leftrightarrow V(x \Leftrightarrow 1, y, z); \\
N_y &= V(x, y+1, z) \Leftrightarrow V(x, y \Leftrightarrow 1, z); \\
N_z &= V(x, y, z+1) \Leftrightarrow V(x, y, z \Leftrightarrow 1);
\end{aligned} \tag{3.5}
$$

$$
\tag{3.6}
$$

3.2.3 Formação da Imagem

Esta etapa do processo de visualização recebe um volume (vetor 3D) de valores escalares (valores de cor e opacidade em cada ponto), e gera uma imagem resultante da projeção e acumulação das contribuições dos valores de cada um dos *voxels*. Para isto, modela-se este processo como uma aproximação do cálculo da propagação de luz através de um meio participativo (volume). O meio pode ser modelado como um bloco de um gel semi-transparente com os valores de cor e opacidade representados a partir das amostras do volume classificado. A luz ao percorrer este meio interage com o gel, podendo ser absorvida, espalhada ou emitida pelo meio (figura 3.7).

Se considerarmos somente estes processos de interação luz/meio, podemos trabalhar com a aproximação da ótica geométrica. Assim, a interação da luz com os elementos de volume pode ser completamente descrita no âmbito da teoria de transporte [Krueger, 1990]. Isto significa que para gerar a imagem devemos integrar continuamente os efeitos das propriedades óticas ao longo de cada raio de visão. O valor do *pixel* (i,j) é gerado pela integração dos efeitos de interação da luz com os *voxels* ao longo do raio de visão r.

Em geral, assume-se o modelo emissão-absorção (*density emitter model*) [Sabella, 1988], no qual o volume é assumido como preenchido por partículas emissoras de luz, controladas por uma função de distribuição de densidade. A aproximação para volumes de baixa reflectância (*low albedo*) pode ser expressa por:

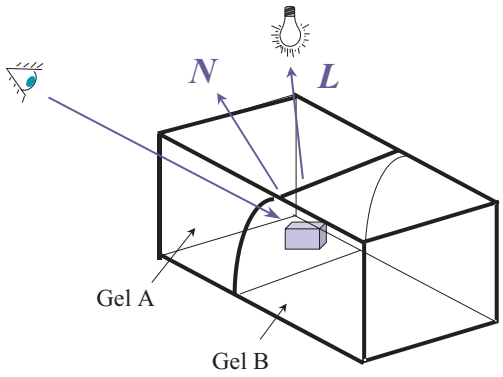

Figura 3.7: Modelo de iluminação no *voxel*

$$I_{i,j} = \int_{t_A}^{t_B} e^{-\int_{t_A}^{t} \tau(s)\,ds} I_V(t)\,dt, \tag{3.7}$$

onde $I_{i,j}$ é a intensidade luminosa do volume transportada na direção do raio associado ao *pixel* (i,j) da imagem; $\tau(s)$ é o coeficiente de atenuação (oclusão) por unidade de comprimento ao longo do raio; e I_v é a intensidade luminosa do volume por unidade de comprimento ao longo do raio.

Esta equação é obtida admitindo as seguintes hipóteses:

- todos os fótons atingindo a imagem foram refletidos apenas uma vez no objeto visível;
- não é considerada a atenuação da luz pelos *voxels* que ficam entre a fonte luminosa e o *voxel* a ser iluminado;
- é considerada a absorção isotrópica.

Para distribuições arbitrárias de I_v e τ, não existe solução analítica para esta integral, o que torna necessário utilizar uma aproximação numérica. Usando uma regra de quadratura para a integral em t (externa) e assumindo que I_V e τ são constantes para um certo segmento i de comprimento do raio, reduzimos a integração a um somatório:

$$I_{i,j} = \sum_{k=0}^{n} I_k \prod_{l=0}^{k-1} e^{\tau_l}, \tag{3.8}$$

onde

$$I_k = I_V \left(\frac{t_k + t_{k+1}}{2} \right) \tag{3.9}$$

$$\tau_l = \tau \left(\frac{t_l + t_{l+1}}{2} \right) \tag{3.10}$$

Definindo $\alpha_l = 1 \Leftrightarrow e^{-\tau_l}$ como a opacidade da amostra l, e $C_k = I_k$ como a cor da amostra, temos:

$$I_{i,j} = \sum_{k=0}^{n-1} (C_k \prod_{l=0}^{k-1} (1 \Leftrightarrow \alpha_l)) \tag{3.11}$$

O operador *over* proposto em [Porter and Duff, 1984], expresso por:

$$over(fg, bg) = \alpha_{fg} fg + (1 \Leftrightarrow \alpha_{fg}) bg, \tag{3.12}$$

simula a composição da cor de dois materiais transparentes de cor fg (frente) e bg (fundo). Utilizando este operador, podemos avaliar este somatório através de operações lineares ao longo do comprimento do raio de visão.

$$\begin{aligned} I_{i,j} &= C_0 + C_1(1 \Leftrightarrow \alpha_0) + C_2(1 \Leftrightarrow \alpha_0)(1 \Leftrightarrow \alpha_1) + \cdots + C_{n-1}(1 \Leftrightarrow \alpha_0) \cdots (1 \Leftrightarrow \alpha_{n-2}) \\ &= C_o \text{ over } C_1 \text{ over } C_2 \text{ over } \cdots \text{ over } C_{n-1}. \end{aligned} \tag{3.13}$$

3.3 Algoritmos

Uma vez obtido o volume classificado e iluminado, o algoritmo fica determinado pelo método utilizado para percorrer o conjunto de dados (*structure traversal*) e gerar a imagem com a projeção adequada. Existem quatro abordagens básicas para o problema de visualização direta de volumes: ordem da imagem (forward mapping), ordem dos objetos (backward mapping), métodos baseados em mudança de base e métodos baseados em transformações geométricas do volume. Na primeira abordagem usualmente lança-se um raio associado a cada *pixel* da imagem e encontra-se os *voxels* interceptados, os quais contribuem para o valor do *pixel* (figura 3.8a). Já a abordagem baseada na ordem dos objetos, percorre o volume, e para cada *voxel* encontra os *pixels* que são afetados pela sua contribuição(figura 3.8b).

O algoritmo de *ray casting*, representante da primeira classe de algoritmos, baseia-se em raios lançados do ponto de vista do observador passando através de cada *pixel* da tela e interceptando o volume. Se o raio interceptar o

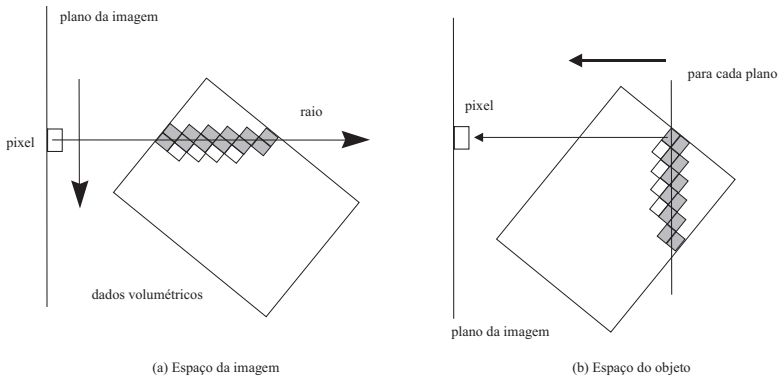

Figura 3.8: Algoritmo do espaço da imagem e do objeto

volume, o conteúdo do volume ao longo do raio é amostrado, transformando-se o valor em cor e opacidade, resultando na contribuição destes elementos de volume que será atribuída ao *pixel* correspondente.

Técnicas da ordem da imagem, possuem como principal vantagem o fato de trabalharem apenas com os elementos do volume que contribuem para a imagem a ser gerada, e a inexistência de representações intermediárias do volume. Além disto, algumas otimizações podem ser implementadas para volumes esparsos de modo a tirar proveito da coerência espacial. Para isto, estruturas de dados do tipo *octree* são usadas, o que acelera consideravelmente o processo de visualização [Fujimoto and Iwata, 1985] e [Levoy, 1990]. Por outro lado, algoritmos da ordem da imagem acessam os dados do volume de forma aleatória, dificultando estratégias de *cache* de dados, e aumentado o tempo de latência da memória. Podemos, afirmar que os principais problemas desta classe de algoritmos estão relacionados à utilização intensiva de memória e ao tempo necessário para gerar a imagem.

O algoritmo *splatting*, que trabalha na ordem dos objetos, projeta os *voxels* no espaço da tela ([Westover, 1990]). Este método necessita que os *voxels* sejam ordenados do mais distante para o mais próximo (*back-to-front*) ou vice-versa (*front-to-back*). Em geral é sacrificada a qualidade da imagem para obter um melhor desempenho, gerando assim imagens de pior qualidade que os métodos do espaço da imagem. No entanto, estes métodos permitem resolver o problema de latência da memória e a implementação de estratégias eficientes de *cache*, pois processam o volume segundo a ordem de armazenamento.

Uma outra classe de algoritmos, mais recente, contém os algoritmos baseados na transformação do volume para um outro espaço (e.g. freqüências) a partir de onde é realizada a visualização. Estes métodos se baseiam na

utilização do teorema da projeção-fatia de Fourier [Cartwright, 1990], modificando consideravelmente o métodos utilizado para percorrrer os dados. Como representantes desta classe temos as propostas de [Malzbender, 1993] e [Totsuka and Levoy, 1993]. Também podem ser incluidos nesta classe os métodos baseados em transformadas de *wavelets* que percorrem uma estrutura hierárquica que representa o volume.

A seguir apresentamos os dois algoritmos básicos para visualização direta de volumes. Inicialmente apresentamos o algoritmo *ray casting*, que gera imagens de muito boa qualidade, necessitando de um considerável esforço de processamento, para em seguida discutir o algoritmo *splatting* [Westover, 1990], ordem dos objetos, muito utilizado em implementações paralelas de visualização de volumes.

3.3.1 *Ray Casting*

O algoritmo *ray casting* percorre todo os *pixels* da imagem determinando a cor e a opacidade de cada um. Para isto, um raio é percorrido partindo de cada *pixel* em direção ao volume. As opacidades e cores encontradas ao longo do raio são acumuladas até se determinar a cor e a opacidade final do *pixel* . Várias otimizações, melhorias e métodos híbridos são citados na literatura, principalmente em [Tuy and Tuy, 1984], [Levoy, 1988], [Levoy, 1990], [Seixas et al., 1994] e [Upson and Keeler, 1988]. Este algoritmo possui alto custo computacional, mas é facilmente paralelizável e permite a exibição de todo o conjunto de dados. O algoritmo 3.1 apresenta em linhas gerais as etapas do algoritmo *ray casting*.

Algoritmo 3.1

```
para i de 1 a ImageWidth
    para j de 1 a ImageHeight
        prevColour = backgroundcolor
        para l de 1 a RayLength
            //calcula proxima posição ao longo do raio
            rayPoint = TraverseData (i, j, viewDirection)
            // acessa os voxels correspondentes para interpolação
            sample = InterpolateVoxelData(rayPoint)
            // calcula a cor do voxel
            color=Shade(sample, normal, lightSource,...)
            // realiza a classificação
            opacity = Classify(sample, normal, transferFunction)
            //etapa de composição
            intensity=Composite(color, prevColor, opacity)
```

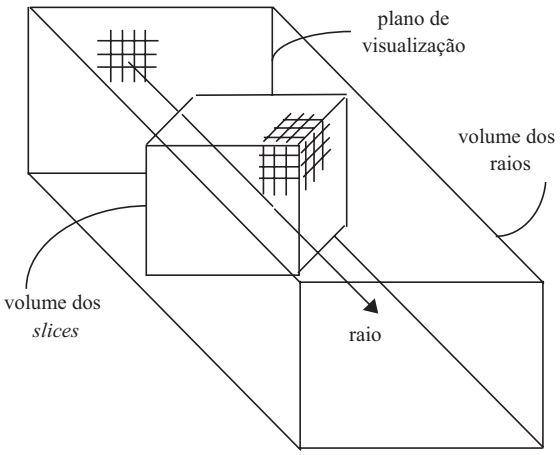

Figura 3.9: Lançamento do raio

A primeira etapa a ser realizada é o lançamento dos raios, para isto podemos calcular a interseção de um raio de visão com o volume de dados e definir o ponto de entrada (t_e) e saída (t_s). Em seguida a interseção pode ser calculada de maneira incremental. Então, é realizada uma amostragem com espaçamento unitário dentro do volume, e para cada amostra é aplicado um filtro de reconstrução (interpolação) para determinar o valor do campo escalar do volume no ponto amostrado. Em cada ponto amostrado são determinadas as propriedades do material representado (cor e opacidade), sendo realizado o cálculo da iluminação. A abordagem mais comun para este processo utiliza uma tabela associando valores escalares do volume a cor e opacidade (Tabela LUT) e utiliza um modelo local de iluminação como o de Phong [Phong, 1975]. Determinados os valores das amostras ao longo do raio, é realizado o processo de composição através da aplicação da equação 3.13.

Algumas técnicas de aceleração foram propostas, todas tentam aproveitar o conhecimento explícito sobre os dados volumétricos durante o processo de integração.

Levoy [Levoy, 1988], introduziu os conceitos básicos de aceleração por presença e terminação precoce do raio (terminação α), que se tornaram comuns nas implementações atuais. A aceleração por presença detecta espaços vazios no volume e não realiza a composição em trechos do raio que cruzam essas subregiões. A terminação α mantém um valor da opacidade acumulada durante o processo de composição, o qual é parado quando este valor se torna suficientemente próximo de 1 (totalmente opaco), desprezando as amostras que se encontram na parte posterior do raio, as quais não irão contribuir para

cor do *pixel* .

Danskin [Danskin and Hanrahan, 1992] extendeu a idéia de aceleração por presença para aceleração por homogeneidade, o que permite percorrer regiões vazias ou homogêneas do volume mais rapidamente. A terminação precoce também foi extendida, para adaptar o passo de amostragem ao longo do raio à opacidade acumulada (aceleração β).

Sakas [Sakas and Gerth,] associou a frequência ao longo do raio às menores frequências presentes na versão passa-baixa em diferentes níveis da pirâmide de multiresolução do volume. Para acelerar o processo de visualização o passo de integração é adaptado à resolução das versões filtradas que serão percorridas.

3.3.2 Splatting

O algoritmo *splatting* [Westover, 1990] é inspirado na estrutura do *pipeline* de *rendering* de polígonos, onde cada primitiva passa ao longo dos vários estágios por vez. Neste algoritmo, cada elemento é mapeado no plano da imagem, e através de um processo de acumulação, tem sua contribuição adicionada à formação da imagem. O algoritmo termina quando todas as primitivas são mapeadas na imagem.

Podemos definir o algoritmo através da aplicação de quatro etapas principais: transformação, classificação/iluminação, reconstrução e visibilidade. O processo de transformação é composto do mapeamento das coordenadas (x, y, z) do volume em coordenadas de visualização (i, j, k). Este mapeamento é realizado através da definição da matriz de projeção ortográfica, sendo efetuado de maneira mais eficiente através de cálculos incrementais.

Após determinada a matriz de transformação para incrementos nas três direções do volume, o algoritmo determina a ordem em que deverá percorrer o volume para manter a ordem de composição desejada. Os dois primeiros eixos que serão percorridos primeiro, formam um plano que será projetado e onde serão acumuladas as contribuições dos *voxels*, este plano forma um *buffer* denominado *sheet*. Assim, são projetados cada um dos *voxels* do volume, os quais são enviados para o processo de classificação/iluminação.

Este processo determina a cor e opacidade de acordo com o valor do *voxel*, e aplica o modelo de iluminação apropriado gerando a cor e a opacidade final de cada *voxel*.

O passo seguinte, central ao algoritmo, é a reconstrução. Neste passo o algoritmo se propõe a reconstruir um sinal contínuo, a partir de um volume discreto, de modo a reamostrar o sinal na resolução desejada para gerar a imagem.

Para isto é criada uma tabela representando a função de *footprint*, a qual é centralizada na projeção de cada *voxel*. Em seguida determina-se os *pixels* cujos centros estão contidos sob a projeção da tabela. Nestas posições pega-se o valor da tabela e pondera-se os valores do *voxel*, somando-o ao valor corrente no *sheet buffer*.

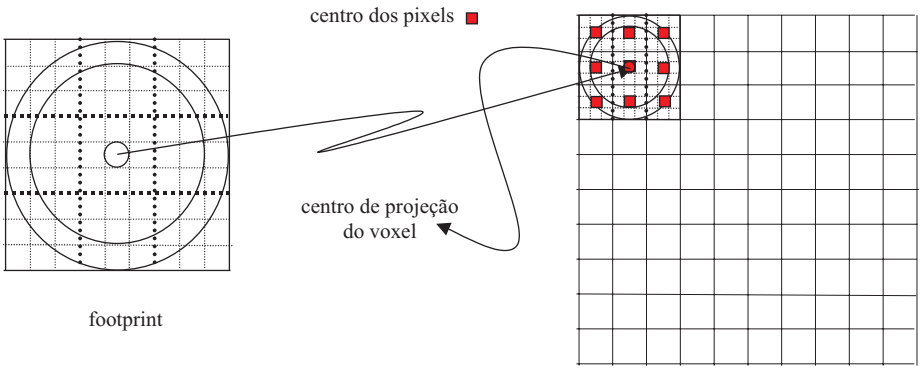

Figura 3.10: Utilização da tabela de *footprint*

A forma mais simples de construir a tabela que representa a função de *footprint* é determinar a extensão da projeção do núcleo do filtro de reconstrução (kernel) sobre o plano da imagem, e selecionar uma resolução maior que a da imagem para a amostrar a função. Em seguida, a integração do núcleo de recosntrução é realizada para cada um dos pontos gerados. Uma forma diferente de realizar este processo é através da modelagem do resultado com uma função simples (e.g. Gaussiana), que é avaliada para cada ponto da tabela.

Para construir a tabela, o algoritmo calcula a extensão da projeção do núcleo de reconstrução e o mapeamento entre ela e a tabela normalizada básica gerada a partir de uma projeção ortográfica, sem operações de escala, da esfera unitária que contém o núcleo.

Para projeção ortográfica e grid igualmente espaçado nas três direções, temos que o núcleo de reconstrução é mapeado em uma esfera. O algoritmo precisa criar uma tabela transformada apenas para levar em consideração as escalas uniformes que a transformação de visualização impôs aos *voxels*. Assim, a extensão da projeção do núcleo é dada nas duas direções por: $ext = 2 * filterwidth * gridscale * viewscale$, e o mapeamento é dado pela razão entre os raios das duas circunferências:

$$mapping = \frac{1}{gridscale * viewscale} \tag{3.14}$$

O processo de visibilidade recebe a cor e opacidade do *voxel* e pondera estes valores para gerar a contribuição em todos os *pixels* que estão sob a extensão do *footprint*. Os valores ponderados são compostos no *buffer* de acumulação, usando o esquema de acumulação apropriado à ordem de caminhamento no volume.

Como reconstrução é um processo aditivo e visibilidade não, ao finalizar a projeção de uma fatia do volume sobre o *sheet buffer*, realiza-se a etapa de composição deste *buffer* com a imagem já existente no *acumulation buffer*.

Capítulo 4

Visualização Volumétrica com Wavelets

4.1 Introdução

A visualização volumétrica envolve a manipulação de grandes quantidades de dados, o que transforma a eficiência dos algoritmos seu objetivo central. Esta eficiência se manifesta em termos de requisitos de memória e tempo de execução. Isto pode ser observado, verificando que o processo de visualização volumétrica está baseado em uma solução precisa da equação de visualização, para o que é necessário a realização de um grande número de cálculos numéricos, resultando em tempos de processamento que impossibilitam a interatividade. Várias técnicas de aceleração foram propostas para os métodos existentes, mas o crescente aumento na resolução dos volumes impossibilita algumas vezes o seu processamento.

Para melhorar a eficiência dos algoritmos de visualização volumétrica com relação à utilização da memória, tem sido desenvolvidas eficientes técnicas de compressão, que permitem a visualização do volume diretamente a partir dos dados comprimidos.

A idéia básica destes métodos é permitir a reconstrução local de pontos no volume, sem precisar descomprimir todo o volume. Assim, é possível resolver a integral de visualização sobre o domínio comprimido, e com isto diminuir os requisitos de memória necessários à execução do algoritmo.

Malzbender [Malzbender, 1993] e Totsuka [Totsuka and Levoy, 1993] desenvolveram as primeiras técnicas nesta direção, baseados na transformada de Fourier do volume. Muraki [Muraki, 1993] acrescentou a idéia de reconstrução local do volume a partir de sua descrição hierárquica e única oferecida pela teoria de *wavelets*. Esta representação permite a descrição do

volume em termos de uma forma básica (sinal de baixa frequência) adicionada de detalhes (alta frequência) espacialmente localizados que variam em sua largura. Explorando as propriedades de localização, podemos obter estratégias eficientes para a compressão, modelagem e visualização dos dados volumétricos. Isto é possível, entre outras razões, por que a representação hierárquica pemite a exploração da coerência no volume.

Os principais benefícios esperados da utilização de estruturas hierárquicas para a representação e visualização de dados volumétricos são a redução da quantidade de dados envolvidos no processo de visualização e uma melhoria na eficiência computacional destes algoritmos.

Entre outros aspectos que tornam vantajosa a utilização da representação em *wavelets* para os dados volumétricos, podemos citar: o controle global ou local do nível de detalhe (LOD) para a visualização e reconstrução do dado(multiresolução), existência de representações analíticas e contínuas dos parâmetros, e métodos de cálculo progressivos e rápidos para estes parâmetros.

A representação em *wavelets* fornece basicamente uma organização em árvore (*octree*) dos coeficientes da transformada, que contém uma versão filtrada por passa-baixa dos dados e um conjunto de informações de detalhe. Como grande parte dos coeficientes são zero ou próximos de zero, podem ser alcançadas altas taxas de compressão. Além disto há a possibilidade de utilização de técnicas de aceleração aproveitando a esparsidade da representação.

Neste capítulo apresentamos as principais técnicas propostas para a visualização de volumes que utilizam a representação do volume na base de *wavelets*. Inicialmente apresentamos a técnica de *ray casting*, e em seguida o algoritmo de *splatting*, ambos baseados em representações por *wavelets* dos dados volumétricos.

4.2 Ray Casting Baseado em Wavelets

O objetivo do algoritmo de visualização em multiresolução é resolver a integral de visualização do volume utilizando uma representação em multiresolução do campo escalar amostrado (dados volumétricos) ([Westerman, 1994] e [Gross et al., 1995]).

O algoritmo de visualização baseado em pirâmides de multiresolução é representado por três passos básicos:

1. representa a função volumétrica f, fornecida pelos dados amostrados, em um novo sistemas de coordenadas (base de *wavelets*), e comprime caso seja desejável;

2. usa esta representação como um esquema de interpolação para aproximar o valor da função f em qualquer ponto x do volume;

3. resolve a integral de visualização usando um algoritmo de *ray casting* .

O primeiro passo, decomposição do volume na base de *wavelets*, é em geral assumido como um passo de pré-processamento, pois assumimos que o volume a ser visualizado já está representado na base de *wavelets*. Esta decomposição pode ser feita, conforme visto no capítulo 2.

O segundo passo consiste na avaliação da função f em um ponto qualquer do espaço através da utilização de sua decomposição em *wavelets*. Iniciando no nível mais refinado de resolução podemos calcular a contribuição de todos os coeficientes de *wavelets* diferentes de zero e adicioná-los à contribuição decorrente da função de escala. Neste *loop* verificamos qual o nível de resolução mais refinado cujos coeficientes de *wavelet* não contribuem para a reconstrução de f. Se um nível $j > 1$ ($j = 1$ sendo o nível mais refinado de resolução) é encontrado, então podemos incrementar o passo de integração por um fator de 2^{j-1}, adaptando assim o passo de integração à informação contida nos coeficientes de *wavelet* . O ponto central do algoritmo é como percorrer a pirâmide de multiresolução e coletar os coeficientes relevantes. No pior caso, quando todos os coeficientes de *wavelet* são diferentes de zero, em torno de uma centena de multiplicações com ponto flutuante é realizada para interpolação de um ponto ao longo do raio.

O último passo do algoritmo consiste na avaliação da integral de visualização, que pode ser realizada por um esquema de *ray casting* . Quase todas as acelerações implementadas para o *ray casting* podem ser utilizadas, tais como aceleração β, corte de frequência, terminação precoce do raio. O único aspecto novo, é o ajuste do tamanho do passo de integração baseado nas informações dos coeficientes de *wavelets*.

Existem duas maneiras básica de trabalhar com estes algoritmos. Na primeira (Transformada RGBA), assume-se que a função f do volume I_V e a atenuação α são fornecidos de maneira independente. Esta abordagem é baseada na geração da decomposição do volume de intensidade luminosa e de opacidade (até 4 volume transformados), os quais são modificados a cada alteração da função de transferência. Esta abordagem é proposta por [Gross et al., 1995] e está representada na figura 4.1

A outra abordagem (Transformada da Densidade) [Westerman, 1994] trabalha somente com a transformação do conjunto de dados inicial, obtendo a cor e opacidade através de tabelas LUT (*lookup table*), que mapeiam os valores da função f em cor e opacidade. Nas seções seguintes apresentaremos estas duas abordagens.

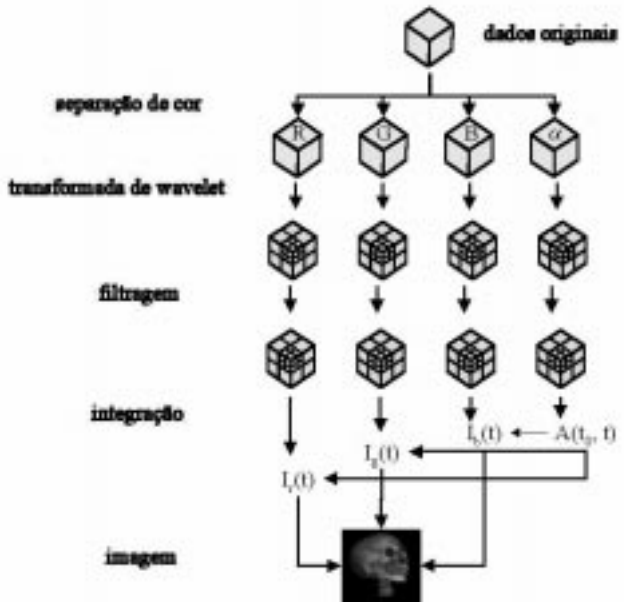

Figura 4.1: Decomposição do volume inicial em 4 canais

4.3 Transformada RGBA

Este método de visualização de volumes baseado em *wavelets* considera que para um volume discreto amostrado em um grid 3D igualmente espaçado, temos: as amostras de cor do volume $c^{q_{rgb}}_{m_0pqr}$, e as amostras de opacidade $c^{\alpha}_{m_0pqr}$. Seja ψ_m a *wavelet* básica na escala m e ϕ_m a função de escala associada. Assim, pela análise de multiresolução podemos construir uma aproximação contínua das amostras através da utilização das funções de escala na resolução $m = m_0$, e geramos os volumes de cor e opacidade por:

$$\alpha^3(x,y,z) = \sum_{pqr \in Z} c^{\alpha}_{m_0pqr}\phi^3_{m_0pqr}(x,y,z)$$
$$q^3_{RGB}(x,y,z) = \sum_{pqr \in Z} c^{q_{RGB}}_{m_0pqr}\phi^3_{m_0pqr}(x,y,z), \tag{4.1}$$

onde

$$\phi^3_{m_0pqr}(x,y,z) = \phi_{mp}(x)\phi_{mq}(y)\phi_{mr}(z) \tag{4.2}$$

e

$$\phi_{mn} = 2^{-m\frac{m}{2}}\phi(2^{-m}x \Leftrightarrow n) \tag{4.3}$$

Se for utilizada a base de splines ([Chui, 1992]), podemos verificar que estas equações correspondem a uma aproximação do volume de dados por B-splines.

Assumimos que existe uma transformada de *wavelets* 3D, como em [Muraki, 1993] e [Gross et al., 1995], que decompõem separadamente cada componente RGB e α de acordo com a figura 4.1

Os coeficientes resultantes representam as coordenadas das funções decompostas no espaço de *wavelets*(W_j^3) e de escala(V_j^3). Como consequência, temos uma aproximação hierárquica das amostras iniciais. Esta expansão em multiresolução é dada por:

$$\alpha^3(x,y,z) = \sum_{m=m_0}^{M} \sum_{type=1}^{7} \sum_{pqr \in Z} d_{mpqr}^{\alpha,type} \psi_{mpqr}^{3,type} + \sum_{pqr \in Z} c_{mpqr}^{\alpha} \phi_{mpqr}^{3}$$

$$q^3(x,y,z) = \sum_{m=m_0}^{M} \sum_{type=1}^{7} \sum_{pqr \in Z} d_{mpqr}^{q_{RGB},type} \psi_{mpqr}^{3,type} + \sum_{pqr \in Z} c_{mpqr}^{q_{RGB}} \phi_{mpqr}^{3} \qquad (4.4)$$

Baseado na expansão das equações 4.4 podemos calcular o gradiente de opacidade $\nabla\alpha$, fundamental para os modelos locais de iluminação. A normal $n(x(t), y(t), z(t))$ definida para uma posição espacial qualquer ao longo de um raio de visão é dada por:

$$n(t) = \frac{\nabla\alpha(t)}{\mid \nabla\alpha(t) \mid} \qquad (4.5)$$

Como α é definido por uma combinação linear de funções de *wavelet* e escala, temos:

$$\frac{\partial\alpha^3(x(t),y(t),z(t))}{\partial x} = \frac{\partial}{\partial x} \sum_{m=m_0+1}^{M} \sum_{type=1}^{7} \sum_{p,q,r \in Z} d_{mpqr}^{\alpha,type} \psi_{mpqr}^{3.type} + \frac{\partial}{\partial x} \sum_{pqr \in Z} c_{Mpqr}^{\alpha} \phi_{Mpqr}^{3} \qquad (4.6)$$

Expressões semelhantes podem ser obtidas para $\frac{\partial\alpha^3}{\partial y}$ e $\frac{\partial\alpha^3}{\partial z}$. Em razão da natureza tensorial das funções da base ϕ^3 e ψ^3, as expressões 4.6 podem ser calculadas analiticamente. Temos que a função de escala satisfaz:

$$\frac{\partial\phi_{mpqr}^3}{\partial x} = (\frac{d\phi_{mp}(x)}{dx}\phi_{mq}(y)\phi_{mr}(z)) \qquad (4.7)$$

e expressões semelhantes podem ser derivadas para as *wavelets*. Se as funções da base possuirem forma analítica (caso das Bsplines), a normal pode ser calculada analiticamente.

Escrevendo a equação de visualização (equação 3.7) como uma fórmula de recorrência temos:

$$I_{i,j}(t_A, t_i) = I(t_A, t_i) + \int_{t_{i-1}}^{t_i} I_V(t) e^{\int_{t_{i-1}}^{t_i} \alpha(s)\, ds}\, dt \tag{4.8}$$

Esta relação resulta em uma discretização do parâmetro ao longo do raio t. Baseados na aproximação de α e q (4.4) devemos encontrar um método de quadratura para resolver a equação 4.8 numericamente.

O ponto central do algoritmo é a reconstrução local. Neste passo, os termos da iluminação e opacidade do volume são compostos, a partir dos coeficientes de *wavelet* . Em razão do suporte compacto das funções da base, é necessário apenas utilizar um limitado número de *wavelets* para avaliar as funções α e q em qualquer posição espacial. Assim o problema central se torna como encontrar eficientemente as funções da base que são relevantes para o ponto a ser reconstruído.

Para isto, podemos definir relevância como o conjunto de todas as funções da base que não se anulam, e.g.

$$rel(f,t) = \{(m,p,q,r) \in Z^4 \mid (x(t), y(t), z(t)) \in supp(f^3_{mpqr})\} \tag{4.9}$$

Uma técnica rápida e precisa para detectar este conjunto é essencial para restringir as somas nas equações 4.4. Em 1D definimos o suporte de uma função de *wavelet* ψ ou de escala ϕ como:

$$[l_\phi, r_\phi] = supp(\phi) \tag{4.10}$$

$$[l_\psi, r_\psi] = supp(\psi) \tag{4.11}$$

Esta expressão generalizada para uma escala m e uma translação p pode ser escrita como:

$$supp(f_{mp}) = [(l_f + p)2^m, (r_f + p)2^m]\,;\; f \in \phi, \psi, m, n \in Z \tag{4.12}$$

Assim, para cada resolução m e uma dada posição $(x, y, z) \in \Re^3$ os parâmetros p, q, e r devem satisfazer:

$$\lfloor \frac{x}{2^m} \Leftrightarrow r_f \rfloor \leq p \leq \lceil \frac{x}{2^m} \Leftrightarrow l_f \rceil \tag{4.13}$$

$$\lfloor \frac{y}{2^m} \Leftrightarrow r_f \rfloor \leq q \leq \lceil \frac{y}{2^m} \Leftrightarrow l_f \rceil \tag{4.14}$$

$$\lfloor \frac{z}{2^m} \Leftrightarrow r_f \rfloor \leq r \leq \lceil \frac{z}{2^m} \Leftrightarrow l_f \rceil \tag{4.15}$$

4.3.1 Estrutura de Dados e Implementação

Em razão da múltipla sobreposição das regiões de suporte das funções de *wavelet* e de escala, é necessário um esquema sofisticado para gerenciar os dados, de modo a garantir um desempenho eficiente.

A implementação proposta em [Gross et al., 1995] utiliza uma estrutura baseada em uma lista de dois níveis para cada raio. Esta lista contém as funções da base que contribuem para a reconstrução dos valores ao longo do raio.

O primeiro nível da lista é organizado de acordo com os diferentes níveis de decomposição. No segundo nível são mantidos todos os coeficientes RGBα não nulos cujo parâmetro de saída t_A é maior que o atual parâmetro t do raio. Estes coeficientes são ordenados em relação a t_A, conforme ilustrado na figura 4.2.

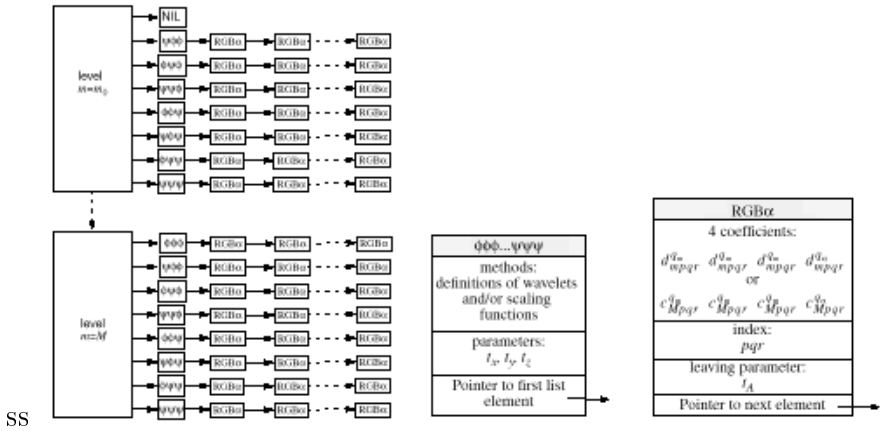

Figura 4.2: Estrutura dos coeficientes de *wavelet* relevantes para um raio

Uma vez que o parâmetro de integração t excede o parâmetro de saída t_A de um elemento da lista ele é removido. Novos elementos são adicionados se t entra na região de suporte de uma nova *wavelet* .

Para uma *wavelet* de um tipo específico e um passo de iteração m, as regiões de suporte não diferem em tamanho e são alinhadas com os eixos principais do sistema de coordenadas. Ou seja, todos os parâmetros de interseção podem ser derivados de um cálculo inicial usando as arestas da função base f_{m000}.

Seja um raio de visão definido por sua origem r_i e pela direção de visualização r_{dir}. Os parâmetros de interseção do raio com as três *leaving edges*

são obtidos por clipping paramétrico:

$$t_{leaving,i} = \begin{cases} \frac{b_i - r_i}{r_{dir,i}} & \text{se } r_{dir,i} > 0 \\ \alpha & \text{se } r_{dir,i} = 0 \\ \frac{a_i - r_i}{r_{dir,i}} & \text{se } r_{dir,i} < 0 \end{cases} \tag{4.16}$$

$$dt_i = \begin{cases} \frac{b_i - a_i}{r_{dir,i}} & \text{se } r_{dir,i} \neq 0 \\ 0 & \text{se } r_{dir,i} = 0 \end{cases} \tag{4.17}$$

,onde a e b determinam a caixa de fronteira (*bounding box*) da funcão da base e $i \in \{x, y, z\}$.

Cada função da base, transladada a partir da função base por p, q, r fornece os seguintes parametros de interseção:

$$t_x = t_{leaving,x} + p\dot{d}t_x t_y = t_{leaving,y} + p\dot{d}t_y t_z = t_{leaving,z} + p\dot{d}t_z \tag{4.18}$$

, e o t_A é dado por: $t_A = minimo(t_x, t_y, t_z)$. Estes cálculos estão ilustrados na figura 4.3 para um caso bidimensional.

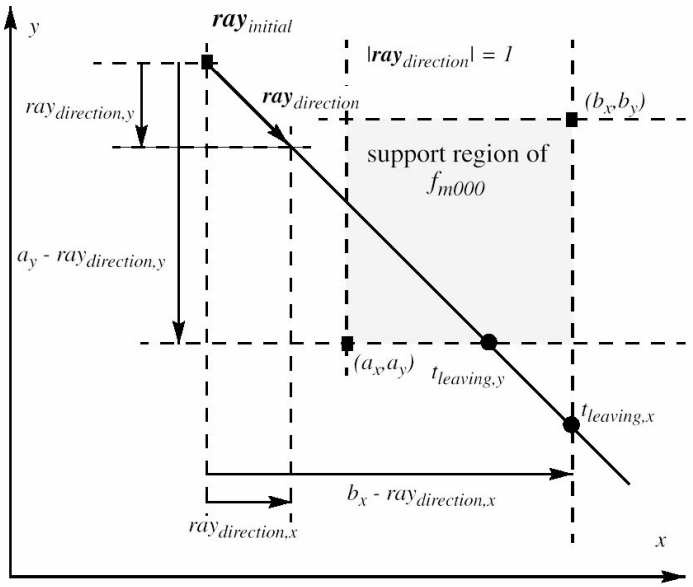

Figura 4.3: Cálculo do parâmetro t_A para uma função da base

4.4 Transformada da Densidade

O algoritmo proposto por Westermann [Westerman, 1994], assume que é possível reconstruir pontos arbitrários entre amostras discretas do volume, o que permite a utilização de uma descrição contínua em todo o intervalo em que as amostras estão definidas.

Considerando que a hierarquia de multiresolução é definida do nível 0 ao nível $\Leftrightarrow$N, então, um ponto x pode ser reconstruído a partir dos coeficientes de *wavelet* q_{ijk}^{ml} de cada nível l e da informação de escala p_0^{-N}, por:

$$f(\overset{\Leftrightarrow}{x}) = \sum_{m=0}^{6} \sum_{l=-N}^{0} \sum_{ijk} q_{ijk}^{m,l} N_{ijk}^{m,l}(x) + \sum_{ijk} p_0^{-N} \phi_{ijk}^{-N}(x) \tag{4.19}$$

Na implementação, estes coeficientes devem ser inspecionados para verificar se contribuem para o ponto a ser reconstruído. Estes coeficientes são encontrados através de uma busca por funções da base que se sobrepõem ao ponto que será reconstruido. No final, as contribuições de todos os espaços de diferença e a informação de escala final devem ser somados como mostra o algoritmo 4.1.

Algoritmo 4.1

```
float volume::pyramide_3D () // volume transformado
Sample X // ponto a ser reconstruido
int   depth = ⇔N
float volume::reconstruct (Sample X, int* depth )
{
  float val = 0
  level = 0 ;
  while(level >= ⇔N ) // passos nos niveis de resolução
  {
    for ( each wavelet ψ^level_ijk)
    {
      val+ = volume.coeff^level_ijk ψ̇^level_ijk(x)
      *depth = max(level, *depth)
    }
    level = level ⇔1
  }
  // adiciona a informação final
  for (each função de escala φ^-N_ijk )
  {
    val+ = volume.coeff_0^-N φ^-N_ijk(x)
```

```
  }
  retorne val
}
```

Já que é possível reconstruir pontos arbitrários dentro do volume, podemos resolver a integral de visualização diretamente sobre a aproximação em multiresolução. Na discussão a seguir vamos ignorar a influência da função de escala.

Representando o volume de dados discretos $f(x)$ em termos das funções da base de *wavelets* e movendo o somatório para fora, temos a seguinte aproximação da integral de visualização:

$$I \approx \sum_{m=0}^{6} \sum_{l=-N}^{0} \sum_{ijk} q_{ijk}^{m,l} \int_{t=t_0}^{t_1} N_{ijk}^{m,l} dt, \tag{4.20}$$

onde a atenuação ao longo do raio foi desprezada. Em cada amostra ao longo do raio devemos reconstruir o valor do campo escalar do volume, e as cores e opacidades correspondentes são determinadas pela LUT.

A equação 4.20 é repetida para cada canal RGBA da descrição do volume. Esta expressão afirma que a precisão do processo de integração é definida pelas escalas das funções da base envolvidas. O algoritmo 4.2 apresenta o núcleo de uma implementação de *ray casting* em um volume representado por multiresolução.

Algoritmo 4.2

```
Sample X
float lenght = volume::intersect(ray, &X)
int   step = 1
rgba ray:mr_trace (Volume volume, float lenght, Sample X)
{
  rgba s, c(0,0,0,0)
  float val
  int depth
  while(lenght > 0 )
  {
    val = volume::reconstruct(X, &depth)
    s = map (val)
    c = over (c,s)
    lenght -= step
    x += step.ray.dir
  }
```

return c
}

Uma das principais vantagens da projeção de *wavelets* advém da propriedade de momentos nulos das funções da base. Em regiões onde os dados são homogêneos um grande número de coeficientes terá valor próximo a zero. Em razão disto podem ser desprezados sem afetar a qualidade visual da imagem. E a avaliação da integral de visualização sobre projeções esparsas em diferentes espaços possui algumas propiedades interessantes:

1. Compressão: como é possível reconstruir pontos arbitrários, não há a necessidade de reconstrução total do volume para gerar a visualização. Consequentemente o volume pode ser reconstruído localmente ao longo dos raios de visão, e somente estes coeficientes devem ser levados em consideração.

2. Aceleração: quanto menor o número de coeficientes que contribuem para a soma da equação 4.20, menor o número de operações de multiplicação que serão efetuadas.

3. Previsão: em razão da integral da equação 4.20 ser realizada ao longo das funções da base, podemos prever a coerência espacial do volume ao longo de um raio.

4.4.1 Acelerações

Este algoritmo permite a implementação de várias técnicas comuns de aceleração no algoritmo de *ray casting* .

A aceleração β pode ser realizada, baseada na detecção de regiões homogêneas ou vazias no volume, o que pode ser feito de maneira implícita na estrutura de multiresolução. Posto que regiões suficientemente homogêneas são representadas nas resoluções menos refinadas, somente funções da base até esta escala são envolvidas no processo de reconstrução. Assim, a homogeneidade da função pode ser estimada a partir do nível de resolução mais refinado que contribui para os valores reconstruídos.

Iniciando no nível de resolução mais refinado da hierarquia de multiresolução, podemos adaptar o passo integração em cada nível subsequente para adaptá-lo à escala das funções da base associadas. Da relação de escala dupla podemos verificar que o passo de integração dobra a cada nível de resolução que descemos. No entanto, em regiões onde o passo de integração é maior que a unidade, deve ser tomado cuidado na aplicação do operador *over*, pois

a opacidade do volume é expressa por unidade de comprimento ao longo do raio.

Observando a função *reconstruct* no algoritmo 4.2, que sempre retorna a escala mais refinada que contribui para o valor a ser reconstruído, podemos adaptar o passo de integração a medida que o caminhamento ao longo do raio é realizado. O passo de integração é inversamente proporcional à escala, devendo ser restaurado em regiões menos homogêneas. O algoritmo 4.3 apresenta a implementação com esta aceleração.

Algoritmo 4.3

```
Sample X
float lenght = volume:intersect(ray, X)
int   step = 1
rgba ray:mrtrace(Volume volume, float lenght, Sample X)
{
  rgba s, c(0,0,0,0)
  int depth, level= 0
  float val
  while(length > 0)
  {
    val = volume:reconstruct(X, depth)
    s = map(val)
    c = over(c,s)[1 << (⇔level)]
    level = depth > level?depth : level
    step =1 << (⇔level)
    X+= step * ray.dir
    lenght -= step
    level -= 1
  }
  return c
}
```

Outra possibilidade de aceleração pode ser implementada observando que o efeito de erros de integração em regiões opacas são de pouca relevância. Assim, o erro decorrente da eliminação de coeficientes de *wavelets* pode ser relacionado à opacidade acumulada ao longo do raio. Esta idéia está apresentada no algoritmo 4.4.

Algoritmo 4.4

```
float Δε !! erro adicional
int   depth = 0
```

```
Sample X !! ponto amostrado
float volume:reconstruct(Sample X, float δϵ, int depth)
{
  float val, error=0.0
  int level = 0
  while(level >= ⇔N) {
        foreach ( wavelet ψ^level_ijk) {
            tmp=volume.coeff^level_ijk ψ^level_ijk(X)
            if (ψ^level_ijk(X) ≠ 0 and (error + tmp ≤ δϵ ) {
                error += tmp
            } else {
                depth = max(level, depth)
                val += tmp
            }
        }
        level -= 1
  }
  for each ( função de escala ) {
        val+ = volume.coeff_0^-N φ^-N_ijk(X)
  }
  return val
}
```

4.4.2 Estruturas de Dados

Westermann propôs duas formas para armazenar e manusear os dados de maneira eficiente. A primeira organiza os coeficientes da transformada em uma *octree*, com o custo de gastar algumas posições de memória para armazenar os ponteiros da estrutura. A outra opção utiliza um vetor de bytes para esta representação.

A Estrutura em Árvore

A decomposição do volume através de convoluções ao longo dos eixos x, y e z, leva naturalmente a uma *octree* como estrutura apropriada para armazenar os coeficientes resultantes. Se o algoritmo inicia na resolução 2^0 e processa até a resolução 2^N, então uma árvore de profundidade $N \Leftrightarrow 1$ é construida. Na raiz da árvore a informação associado ao espaço de escala p_0 e os sete coeficientes de *wavelet* $q^{0,0}_{i,j,k} \cdots q^{6,0}_{i,j,k}$ são armazenados. Em cada nível J de um até $N \Leftrightarrow 2$, cada nó é composto por 7 coeficientes de *wavelets* $q^{0,J}_{i,j,k} \cdots q^{6,J}_{i,j,k}$ e no máximo 8 ponteiros para os nós do próximo nível. No nível final, $N \Leftrightarrow 1$,

as folhas são construídas sem *links* adicionais para outros nós, possuindo somente os 7 coeficientes de *wavelet* (figura 4.4). Além destes coeficientes, em cada nível armazena-se uma máscara de 8 bits para identificar os coeficientes armazenados.

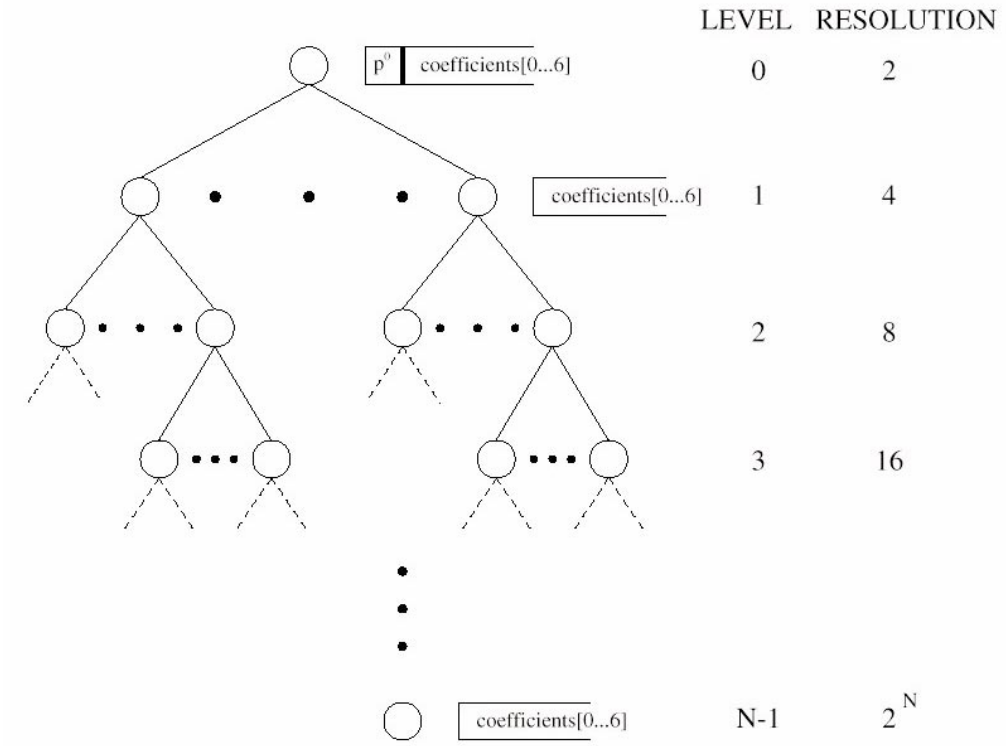

Figura 4.4: Representação em árvore da estrutura de multiresolução do volume.

A vantagem desta estrutura aparece quando coeficientes de *wavelets* são descartados em níveis de maior resolução. Devido à representação esparsa, ramos completos da árvore serão vazios e poderão ser removidos. Isto minimiza a quantidade de memória necessária e acelera o algoritmo de visualização. Além do que, a estrutura de subdivisão permite encontrar facilmente os nós que precisam ser inspecionados.

Estrutura em Vetor

Para evitar a utilização de ponteiros e diminuir a quantidade de memória necessária, podemos armazenar os coeficientes em uma *octree* sem ponteiros. Esta abordagem não permite a economia de memória quando um conjunto

de coeficientes for descartado. A estrutura proposta é um vetor de bytes com coeficientes nulos codificados via *run-lenght*. Para cada nível da estrutura de multiescala um vetor é construido. Assim, coeficientes de uma certa escala J são diretamente acessados. Todos os conjuntos de 7 coeficientes $q_{i,j,k}^{0,J} \cdots q_{i,j,k}^{6,J}$ são armazenados na ordem x, y, z com uma máscara de guia, que define quais são os coeficientes presentes. A principal desvantagem desta estrutura é o tempo de busca para coeficientes arbitrários, necessitando percorrer a estrutura várias vezes para reconstruir um ponto.

4.5 *Splatting* em Multiresolução

Um outro algoritmo para visualização de volumes baseados em multiresolução, foi proposto por [Gross et al., 1997]. Este algoritmo usa a técnica de *splatting* proposta por [Westover, 1990].

O algoritmo proposto, se baseia na idéia de que em muitas aplicações, e.g. imagens médicas, imagens como as produzidas por exames de raios X são satisfatórias. Estas imagens podem ser obtidas através de uma simplificação da integral de visualização que pode ser expressa por:

$$I_{rgb}(\mu,\nu) = \int_{-\infty}^{\infty} q(x(t,\mu,\nu), y(t,\mu,\nu), z(t,\mu,\nu))dt, \tag{4.21}$$

introduzindo a aproximação do volume por *wavelets* temos:

$$\begin{aligned} I_{rgb}(\mu,\nu) = & \sum_{pqr} c_{M,pqr}^{q_{rgb}} \int_{-\infty}^{\infty} \phi_3^{M,pqr}(x(t,\mu,\nu), y(t,\mu,\nu), z(t,\mu,\nu))dt + \\ & \sum_{m=m_0+1}^{M} \sum_{type=1}^{7} \sum_{pqr} d_{m,pqr}^{q_{rgb},type} \int_{-\infty}^{\infty} \psi_{3,type}^{M,pqr}(x(t,\mu,\nu), y(t,\mu,\nu), z(t,\mu,\nu))dt \end{aligned} \tag{4.22}$$

Esta equação pode ser interpretada como uma acumulação ponderada de integrais do tipo $\int_{-\infty}^{\infty} \phi_{m,pqr}^3 dt$ representando as funções de escala e para as *wavelets*: $\int_{-\infty}^{\infty} \psi_{m,pqr}^{3,type} dt$. Estas integrais podem ser compreendidas como *splats* das amostras, em termos de texturas $RGB\alpha$ ponderadas pelos coeficientes da transformada associados. A similaridade das funções da base permitem que as calculemos uma vez para cada uma das 8 funções protótipos $\phi_{M,000}^3$ e $\psi_{M,000}^{3,type}$. Todas as outras texturas podem ser derivadas destas através de escalas e translações no plano da imagem. Este método transforma a visualização de volumes em um processo de acumulação de versões transladadas e escaladas das texturas RGB.

O cálculo das integrais de linha podem ser realizados através da aplicação do teorema da Projeção-fatia de Fourier [Cartwright, 1990]. Este teorema

afirma que a transformada de Fourier de um conjunto de integrais de linha de uma função $f(x, y, z)$ podem ser obtidas através de uma fatia da transformada de Fourier $F(w_1, w_2, w_3)$, em um plano perpendicular às linhas de integração e que passa pela origem.

Como algumas *wavelets*, e.g. Bspline, possuem uma forma fechada para sua transformada de Fourier, a aplicação do teorema pode permitir o cálculo direto dos *splats*. A figura 4.5 apresenta a aplicação deste teorema para uma *wavelet* de Shannon.

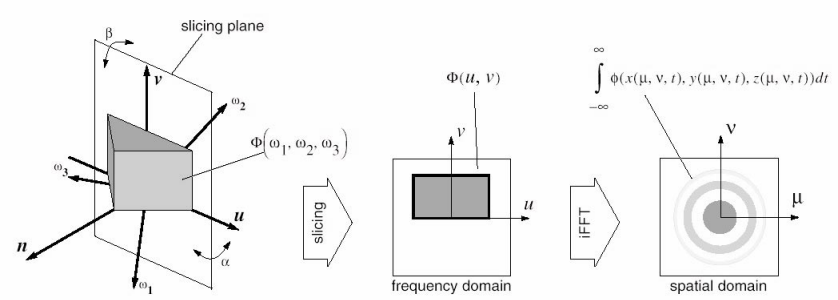

Figura 4.5: Aplicação do Teorema da Projeção de Fourier

Uma vez determinada a tabela de *footprint* através de uma amostragem apropriada dos *splats*, o algoritmo se processa como proposto por Westover ([Westover, 1990]), usando os coeficientes da transformada como fatores de ponderação.

Capítulo 5

Conclusão

Neste trabalho foram apresentados alguns métodos para a aproximação da equação de visualização de volumes, tirando partido da representação hierárquica e esparsa fornecida pela transformada de *wavelets* do dado volumétrico.

Esta abordagem pode ser considerada como a principal abordagem utilizada pelos métodos de visualização volumétrica baseados no domínio da compressão. Isto por que permite a utilização das ténicas básicas de aceleração dos algoritmos de visualização volumétrica, além de possibilitar uma fácil implementação de técnicas de extração de características do volume (segmentação).

Os resultados experimentais apresentados na literatura ([Lippert, 1998] e [Westerman, 1994]) demonstram que os conjuntos de dados volumétricos podem ser visualizados com um tempo de processamento aceitável, com a utilização de apenas um pequeno subconjunto dos dados originais, preservando a informação de alta frequência.

Nas implementações apresentadas do algoritmo de *ray casting*, podemos observar que há um aproveitamento da representação esparsa do volume no domínio de *wavelets*, que resulta do potencial de compressão das funções da base. No entanto, o tempo de processamento é fortemente dependente do suporte das funções da base utilizada. Quanto maior o número de momentos nulos da *wavelet* mãe, mais esparsa é a representação e consequentemente mais trabalho é dispendido durante a integração.

No algoritmo proposto por [Lippert, 1998], a utilização da base de B-Splines permite uma reconstrução precisa dos dados e principalmente, uma estimativa do gradiente, essencial para o cálculo da iluminação. Estes métodos permitem também o controle do erro de reconstrução localmente, fornecendo ao usuário uma maneira de controlar a precisão da imagem e decidir acerca da qualidade da imagem e do tempo de processamento necessário para gerá-la com a precisão desejada.

O algoritmo de *splatting* baseado em *wavelet* , se beneficia da combinação da representação de *wavelets* com a utilização do teorema da projeção-fatia de Fourier. Através da utilização deste teorema são calculados os *footprints* das *wavelets* de maneira rápida e precisa.

As principais desvantagens destes métodos estão relacionadas à eficiência. Somente para representações muito esparsas temos tempo de processamento próximo dos obtidos pelos métodos tradicionais. Se o suporte das funções da base de *wavelet* crescem, então os tempos aumentam consideravelmente. Isto limita a utilização de *wavelets* com grande número de momentos nulos e consequentemente limita a capacidade de compressão da transformada.

Como em quase todas as aplicações de *wavelet* , o problema central é a escolha de uma base de funções que trate esta questão da maneira mais próxima da ótima, possibilitando altas taxas de compressão e tempos aceitáveis de processamento.

Como problemas ainda em aberto, podemos citar:

1. não existência de uma função suave com suporte pequeno e compacto;

2. estudo da transformada de *wavelet* em dados espalhados (*scattered data*), o que pode permitir o desenvolvimento de estratégias hierárquicas eficientes para a visualização de volumes;

3. o desenvolvimento de uma estratégia combinada (ordem da imagem–ordem dos objetos) para a visualização;

4. desenvolvimento de estruturas de dados eficientes e flexíveis para o algoritmo de visualização.

Bibliografia

[Cartwright, 1990] Cartwright, M. (1990). *Fourier Methods for Mathematicians Scietists and Engineers*. Ellis Horwood, New York.

[Castleman, 1996] Castleman, K. (1996). *Digital Image Processing*. Prentice Hall, New Jersey.

[Chui, 1992] Chui, C. K. (1992). *An Introduction to Wavelets*. Academic Press, San Diego.

[Chui and Wang, 1991] Chui, C. K. and Wang, J. Z. (1991). A cardinal spline approach to wavelets. *Proceedings of the American Mathemathics Society*, (113):785–793.

[Cignoni and Scopigno, 1995] Cignoni, P., M. C. P. E. and Scopigno, R. (1995). Multiresolution modeling and visualization of volume data. Technical Report C95-22, INRIA, College Park, MD.

[Danskin and Hanrahan, 1992] Danskin, J. and Hanrahan, P. (1992). Fast algorithms for volume rendering. In *ACM Workshop on Volume Visualization 1992*.

[Daubechies, 1988] Daubechies, I. (1988). Orthonormal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, (41):909–996.

[Daubechies, 1992] Daubechies, I. (1992). *Ten Lectures on Wavelets*. SIAM Books, Philadelphia,PA.

[Daubechies and Lagarias, 1991] Daubechies, I. and Lagarias, J. C. (1991). Two scale difference equation : Existence and global regularity of solutions. *SIAM Journal on Mathematics Analysis*, 22(5):1388–1410.

[Fujimoto and Iwata, 1985] Fujimoto, A. and Iwata, K. (1985). Accelerated ray tracing. In *Computer Graphics: Visual Technology and Art (Proceedings of Computer Graphics Tokyo'85*.

[Gabor, 1946] Gabor, D. (1946). Theory of communication. *Journal of the IEEE*, pages 429–457.

[Gomes and Velho,] Gomes, J. and Velho, L. Abstract paradigms for computer graphics. *The Visual Computer*, 11:227–239.

[Gomes and Velho, 1998a] Gomes, J. and Velho, L. (1998a). From fourier analysis to wavelets. In *SIGGRAPH'98, Course Notes*.

[Gomes and Velho, 1998b] Gomes, J. and Velho, L. (1998b). From fourier analysis to wavelets. SIGGRPAH'98 Course notes.

[Gouraud, 1971] Gouraud, H. (1971). Continuous shading of curved surfaces. *IEEE Transactions of Computers*, 20(6):623–629.

[Gross et al., 1997] Gross, M. H., Lippert, L., Dittrich, R., and Haring, S. (1997). Two methods for wavelet-based volume rendering. *Computers & Graphics*, 21(2):237–252.

[Gross et al., 1995] Gross, M. H., Lippert, L., Dreger, A., and Koch, R. (1995). A new method to aproximate the volume rendering equation using wavelets bases and piecewise polynomials. *Computer Graphics*, 19(1):47–62.

[Grossman and Morlet, 1984] Grossman, A. and Morlet, J. (1984). Decomposition of hardy functions into square integrable wavelets for constant shape. *SIAM Journal of Applied Mathematics*, (15):723–736.

[Krueger, 1990] Krueger, W. (1990). The application of transport theory to visualization of 3d scalar data fields. In *IEEE Visualization'90, Conference Proceedings*, pages 12–15.

[Levoy, 1988] Levoy, M. (1988). Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 5(3):29–37.

[Levoy, 1990] Levoy, M. (1990). Efficient ray-tracing of volume visualization algorithms. *ACM Transactions on Graphics*, 9(3):245–261.

[Lippert, 1998] Lippert, L. (1998). *Wavelet-Based Volume Rendering*. PhD thesis, Swiss Federal Institute of Technology Zurich.

[Mallat, 1989] Mallat, S. (1989). Multifrequency channel decomposition of images and wavelet models. *IEEE Transactions on Acustic, Speech, and Signal Processing*, 37(12):2091–2110.

[Malzbender, 1993] Malzbender, T. (1993). Fourier volume rendering. *ACM Transactions on Graphics*, 12(3):233–250.

[Mccormick et al., 1987] Mccormick, B., Defanti, T., and Brown, M. (November, 1987). Visualization in scientific computing. *Computer Graphics*, . 21(No. 6).

[Muraki, 1993] Muraki, S. (1993). Volume data and wavelet transform. *IEEE Computer Graphics and Applications*, 13(4):50–56.

[Muraki, 1992] Muraki, S. (October 1992). Aproximation and rendering of volume data using wavelet data fields. In *IEEE Visualization'92 Conference Proceedings*, pages 21–28.

[Phong, 1975] Phong, B. T. (1975). Illumination for computer generated pictures". *Communications of the ACM*, 18(6):311–317.

[Porter and Duff, 1984] Porter, T. and Duff, T. (1984). Compositing digital images. *Computer Graphics*, 18(3):235–259.

[Sabella, 1988] Sabella, P. (1988). A rendering algorithm for visualizing 3d scalar fields. *Computer Graphics*, 22(4):51–58.

[Sakas and Gerth,] Sakas, G. and Gerth, M. Sampling an anti-aliasing of discrete 3d volume desnity textures. In *EUROGRAPHICS'91*, pages 87–102.

[Seixas et al., 1994] Seixas, R. B., Gattass, M., Figueiredo, L., and Martha, L. (1994). Otimização do algoritmo de ray-casting na visualização de tomografias. In *SIBGRAPI'94*.

[Sévenier, 1994] Sévenier, M. B. (1994). Réalisation d'une bibliotheque c de fonctions ondelettes. Technical Report 2362, INRIA.

[Totsuka and Levoy, 1993] Totsuka, T. and Levoy, M. (August, 1993). Frequency domain volume rendering. *Computer Graphics*, 27:271–278.

[Tuy and Tuy, 1984] Tuy, H. and Tuy, L. (1984). Direct 2d display of 3d objects. *IEEE Computer Graphics and Applications*, 4(10):29–33.

[Upson and Keeler, 1988] Upson, C. and Keeler, M. (1988). V-buffer - visible volume rendering. *Computer Graphics*, 22(4).

[Wernecke, 1994] Wernecke, J. (1994). *The Inventor Mentor: Programming Object-oriented 3D GRaphics with Open Inventor*. Addison Wesley.

[Westerman, 1994] Westerman, R. (1994). "a multiresolution framework for volume rendering". In *1994 Symposium on Volume Visualization*, pages 51–58.

[Westover, 1990] Westover, L. (1990). Footprint evaluation for volume rendering. *Computer Graphics (Proc. SIGGRAPH)*, 24(4):367–376.

[Williams, 1993] Williams, P. L. (1993). *Interactive Direct Volume Rendering of Curvilinear and Unstructured Data*. PhD thesis, University of Illinois at Urbana-Champaign.