Visualização Volumétrica

Isabel Harb Manssour [1]
Carla Maria Dal Sasso Freitas [2]

Resumo: Visualização Volumétrica denota o conjunto de técnicas relacionadas com a apresentação de informações obtidas a partir volumes de dados complexos, fornecendo, em diferentes graus, possibilidade de exibir e explorar o interior desses volumes. Novas formas de incorporar às técnicas de visualização capacidade de "ver através" dos volumes e de gerar imagens em tempo real tem sido frequentemente relatadas na literatura, como é o caso da utilização recente do mapeamento de textura por hardware. Este artigo apresenta uma introdução à visualização volumétrica abordando, de forma sistematizada, seus aspectos essenciais, quais sejam, a exploração e a exibição do interior dos volumes. São ainda descritos os principais algoritmos para visualização direta de volumes e são identificadas as recentes tendências no desenvolvimento de novas técnicas de exploração de dados volumétricos.

Palavras-chave: Visualização Volumétrica, Ferramentas de corte, Funções de Transferência.

Abstract: Volume visualization techniques are related with the presentation of meaningful information from complex 3-D data sets, providing different ways to exhibit and explore the interior of volume data. New approaches to incorporate seeing-through capabilities into volume rendering techniques and to generate images in real-time have been often described in the literature, as the recent use of the texture mapping subsystem found in modern graphics boards. This paper is an introduction to volume visualization, and presents its core issues, i.e. the means by which we explore and exhibit the structures that are inside the volume. We also describe the main direct volume visualization algorithms, as well as current trends in the development of new techniques for the exploration of volume data.

Keywords: Volume Visualization, Cutting Tools, Transfer Functions.

[1]PUCRS, Faculdade de Informática, Av. Ipiranga 6681, P. 30, 90619-900 Porto Alegre/RS, Brasil
e-mail: {manssour}@inf.pucrs.br
[2]UFRGS, Instituto de Informática, Caixa Postal 15064, 91501-970 Porto Alegre/RS, Brasil
e-mail: {carla}@inf.ufrgs.br
Este trabalho foi parcialmente financiado pela CAPES, CNPq e FAPERGS.

1 Introdução

Este trabalho é uma introdução à Visualização Volumétrica, abordando, essencialmente, técnicas para visualização direta de volumes e apresentando numa abordagem unificada as diferentes formas de explorar e exibir dados volumétricos, manipulando tanto parâmetros geométricos e semânticos como visuais. Esta seção contém uma breve introdução à área identificando algumas de suas aplicações (seção 1.1), a classificação das suas técnicas (seção 1.2) e as abordagens mais recentes, baseadas em hardware (seção 1.3).

O restante do artigo apresenta uma descrição dos principais algoritmos (seção 2) e uma revisão das diferentes formas de explorar o interior dos volumes através de ferramentas de corte (seção 3) e de exibir esse interior através de diferentes funções de transferência de opacidade e cor (seção 4).

1.1 Requisitos e Aplicações

A visualização volumétrica, que emergiu na década de 1990, denota o conjunto de técnicas de Computação Gráfica interativa relacionadas com a apresentação de informações de volumes de dados complexos [13]. O desenvolvimento de aplicações de visualização visa fornecer aos usuários cientistas ferramentas que auxiliem nas mais variadas tarefas que requerem formas de analisar, exibir e explorar o interior de grandes volumes de dados multidimensionais (que geralmente variam com o tempo), para permitir que o usuário consiga identificar características significativas e obter os resultados desejados mais fácil e rapidamente.

Grandes volumes de dados podem ser obtidos por fontes como supercomputadores, satélites de observação da Terra e instrumentos médicos, como, por exemplo, o tomógrafo. As informações normalmente são armazenadas em uma grade regular, mas grades retilíneas, curvilíneas e irregulares também podem ser utilizadas [4]. Entre as várias áreas do conhecimento que se beneficiam das aplicações de visualização volumétrica é possível destacar a medicina, geologia, meteorologia e bioquímica. Como as diferentes modalidades de aquisição de dados médicos geram conjuntos de dados tridimensionais, vários sistemas de visualização e exploração destes dados são desenvolvidos, tanto para auxiliar no diagnóstico médico, como para realizar uma cirurgia assistida por computador ou simulação de cirurgia. Na geologia, as imagens geradas possibilitam ao geólogo uma visão tridimensional da geologia de subsolo, e um melhor entendimento dos dados e suas correlações espaciais. Gráficos tridimensionais e animações que mostram o comportamento atmosférico, são exemplos de aplicações meteorológicas. Na bioquímica, por exemplo, tem se popularizado o uso de computadores para modelar e animar estruturas moleculares, facilitando, assim, a visualização da interação entre as moléculas.

Ferramentas de visualização volumétrica beneficiam-se da disponibilidade de modernas estações de trabalho e computadores pessoais com um bom desempenho, grande quantidade de armazenamento tanto em disco como em memória, e facilidades gráficas. Um aspecto essencial dessas ferramentas é o seu grau de interação na definição e manipulação de parâmetros de visualização. A utilização de equipamentos de realidade virtual que possibilitam a simulação de procedimentos mais próximos da realidade, onde há uma maior interação do usuário, tem também possibilitado avanços consideráveis no uso de técnicas de visualização, particularmente na Medicina.

1.2 Técnicas de Visualização Volumétrica

Na literatura vários termos e expressões são utilizados para caracterizar as diferentes classes de técnicas de visualização de volumes. Por exemplo, Elvins [4] considera duas categorias de algoritmos, *direct volume rendering* e *surface-fitting*, enquanto Müeller et al. [27] reconhece essas duas classes como dois grupos de algoritmos *direct volume rendering* e *indirect volume rendering*. Entretanto, apesar das diferentes classificações e nomenclaturas, as duas categorias de técnicas de visualização de volumes se traduzem nas que trabalham com a extração de uma isosuperfície representada através de primitivas gráficas, e nas que trabalham gerando a imagem diretamente a partir do volume.

Neste trabalho serão utilizados os termos definidos por Kaufman, que classifica as técnicas de visualização de volumes em dois grandes grupos: ***surface rendering*** (ou "visualização através de superfícies") e ***volume rendering*** (traduzido aqui para "visualização direta de volume") [12]. Este autor ainda coloca que a visualização direta de volumes pode ser realizada de três maneiras diferentes [12]: *object-order* (ou *forward-projection*), que envolve o mapeamento de amostras de dados no plano da imagem; *image-order* (ou *backward-projection*), que determina para cada pixel do plano da imagem quais são as amostras que contribuem no cálculo da sua intensidade; e *domain-based*, quando os dados 3D são transformados para outro domínio, como freqüência ou *wavelet*.

Técnicas de visualização através de superfícies envolvem a extração e a representação de uma isosuperfície que é posteriormente visualizada através da utilização de técnicas convencionais da Computação Gráfica. A reconstrução da isosuperfície pode ser feita a partir de contornos planares, como mostra o trabalho de Weinstein [38]. Neste caso, considerando um volume de dados de CT (Tomografia Computadorizada) ou MRI (Ressonância Magnética), a estrutura de interesse deve ser identificada (ou segmentada) em cada uma das fatias para posterior composição da malha de polígonos. A isosuperfície também pode ser definida através de um limiar (*threshold*) [4] ou através de uma malha de polígonos extraída diretamente de um volume 3D previamente segmentado. Durante a etapa de visualização o usuário fornece parâmetros que estabelecem o tipo e a direção de projeção e os parâmetros de iluminação.

Entre os algoritmos de visualização através de superfícies destacam-se: conexão de contornos [7, 14] e cubos marchantes [24]. Alguns dos problemas que devem ser tratados nestes algoritmos são: a geração ocasional de pedaços de superfícies falsos; a manipulação incorreta de pequenas características dos dados; e a dificuldade de representação da superfície de algumas estruturas (particularmente na visualização do corpo humano, no caso de imagens médicas). As grandes vantagens desta técnica, entretanto, são a velocidade para a geração e exibição da imagem final e o pouco espaço de armazenamento requerido. Representações deste tipo são apropriadas quando existem isosuperfícies bem definidas nos dados, mas não são eficientes quando o volume é composto por muitas microestruturas, tais como os tecidos em muitas imagens médicas [27].

A segunda classe, visualização direta de volume, consiste em representar o volume através de voxels 3D que são projetados diretamente em pixels 2D e armazenados como uma imagem, dispensando o uso de primitivas geométricas. Neste caso, numa etapa de classificação, é usada uma função de transferência, que corresponde ao mapeamento dos valores dos voxels (densidade do tecido, por exemplo) para propriedades visuais, tais como cor e opacidade. A visualização das estruturas de interesse dentro do volume é realizada a partir da "visita" a todos os voxels (ou quase todos, dependendo do algoritmo) e da aplicação da função de transferência para a construção da imagem. Estas técnicas possuem um alto custo computacional, pois normalmente envolvem um tipo de interpolação nos pontos ao longo da direção de visualização. Por outro lado, produzem imagens de excelente qualidade, uma vez que todos os voxels podem ser usados na síntese das imagens, possibilitando a visualização do interior dos objetos. Os algoritmos que fazem parte deste grupo são *ray casting* [20, 21] (seção 2.1), *splatting* [39, 40], *shear-warp* [19], *shell rendering* [36], *cell-projection* [42] e *V-Buffer* [37].

Alguns autores [27, 26] fazem uma distinção entre os algoritmos de visualização direta de volume de acordo com a natureza do valor interpolado, que é dependente da ordem das etapas de visualização. Neste caso, quando os voxels do volume de dados são classificados e iluminados como um passo de pré-processamento e depois é realizada a interpolação nos pontos ao longo da direção de visualização, o modo de visualização é conhecido por *pre-shaded volume rendering*. O processo no qual os valores originais dos voxels são interpolados para depois ser feita a classificação e iluminação é denominado *post-shaded volume rendering*.

1.3 *Hardware* Especial para Visualização Volumétrica

Desde a proposição dos primeiros algoritmos, uma das grandes dificuldades no desenvolvimento e utilização de sistemas de visualização interativa de dados volumétricos, principalmente quando se trabalha com técnicas de visualização direta de volume, é a quantidade de dados envolvida em relação às limitações na capacidade de armazenamento e processamento

dos computadores. Apesar do desenvolvimento de algoritmos otimizados, apenas o avanço das tecnologias de hardware nos últimos anos possibilitou uma grande expansão na pesquisa e utilização destas aplicações. Neste caso, três estratégias diferentes podem ser adotadas: utilização de computadores com arquitetura paralela, aproveitamento das placas aceleradoras gráficas e desenvolvimento de hardware dedicado.

Vários algoritmos de visualização, tais como *ray casting* e *splatting*, que possuem um alto custo computacional, são fáceis de paralelizar. Portanto, a utilização de computadores com arquitetura paralela, ou a execução dos sistemas em vários computadores ligados em rede, forçando a implementação do programa de forma distribuída [44], é uma alternativa para reduzir o tempo de geração das imagens. Assim, torna-se possível manipular interativamente os parâmetros de visualização, bem como todo o volume de dados.

Recentemente tem se popularizado o uso de mapeamento de textura 3D por hardware para a visualização de volumes. Novas técnicas de *rendering* são desenvolvidas para tirar vantagem dessa característica disponível nas placas aceleradoras gráficas, incluindo a utilização de multi-texturas [25, 33, 5]. Neste caso, o volume pode ser armazenado como uma textura 3D e o hardware varre fatias poligonais paralelas ao plano de visualização. Estas fatias são então combinadas, por exemplo, de trás para frente usando um filtro de interpolação trilinear [25], resultando numa visualização do volume com transparência controlada.

Outra estratégia adotada para possibilitar visualização interativa é a utilização de hardware dedicado para a visualização de volumes. Kaufman et al. [11] e Ray et al. [32] fizeram uma revisão das arquiteturas existentes para visualização volumétrica. Entre os trabalhos relacionados, destacam-se o *EM-Cube* e o *VolumePro*. O primeiro, *EM-Cube*, é um exemplo de arquitetura VLSI, onde oito *pipelines* implementam o algoritmo de *ray casting* com projeção paralela para visualização de volumes com baixo custo e alta qualidade [29].

VolumePro consiste no primeiro chip para computadores pessoais que permite o *rendering* de um volume com ajuste dos parâmetros de visualização em tempo real [31]. Este chip também implementa o algoritmo de *ray casting* com processamento paralelo e interpolação trilinear, e realizando a estimativa do gradiente, a classificação e a iluminação através do modelo de Phong. Seleção de subvolumes e a utilização de múltiplos planos de corte são outras características avançadas desta arquitetura.

2 Visualização Direta de Volume

Nas seções 2.1, 2.2 e 2.3 são descritos os algoritmos de visualização direta de volume mais populares. Uma breve revisão das técnicas existentes para visualização de estruturas internas é feita na seção 2.4.

2.1 Algoritmo de *Ray Casting*

Resumidamente, o funcionamento do *ray casting* [20, 21] consiste na varredura dos pixels da janela de visualização e no lançamento de raios, a partir desses pixels, na direção do volume de dados (*object-order*). Sendo assim, os principais elementos são o volume de dados e suas propriedades, os parâmetros de visualização, a janela de visualização e os raios ao longo dos quais são processados os pontos de amostragem.

O volume de dados $V(\mathbf{x})$ refere-se a valores **v** associados a posições de amostragem discretas $\mathbf{x} = [x, y, z]$, em geral regularmente espaçadas no domínio 3D, sendo que $x \in [1, N_x]$, $y \in [1, N_y]$ e $z \in [1, N_z]$. A existência de um ou mais valores **v** associados a uma posição de amostragem depende das propriedades dos dados que o volume (ou volumes) $V(\mathbf{x})$ representa. Por exemplo, em um volume de dados de CT, há apenas um valor de densidade. No caso de dados meteorológicos é comum haver mais de um valor escalar associados à mesma posição **x**, tais como temperatura, pressão e umidade relativa do ar.

Os parâmetros de visualização tradicionalmente especificam a posição do observador, o tipo, a direção e o plano de projeção, e os planos de corte 3D. No contexto da visualização direta de volume, a direção de projeção $\overrightarrow{DOP}$ determina a orientação dos raios lançados no volume. O plano de projeção é mapeado para uma janela de visualização I no plano da imagem, que refere-se aos valores de intensidade associados às posições dos pixels $\mathbf{u} = [u, v]$, com $u \in [1, N_u]$ e $v \in [1, N_v]$ (figura 1).

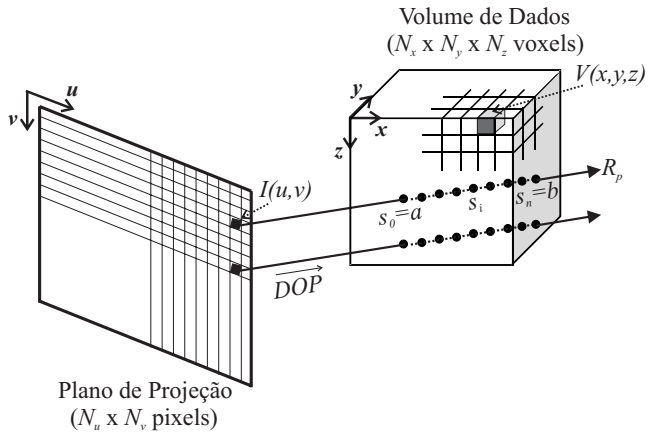

Figura 1. Esquema genérico do algoritmo de *ray casting* (*front-to-back*)

A intensidade para cada pixel da imagem $I(\mathbf{u})$ é determinada pelo lançamento de um raio $\mathcal{R}_p$ deste pixel ao longo da direção de visualização $\overrightarrow{DOP}$ (figura 1). Ao longo de cada

raio são processadas amostras s, em geral com um espaçamento constante. O processamento de uma amostra corresponde ao cálculo de um valor de cor e um valor de opacidade dependentes de **v**. Caso a posição da amostra não coincida com as posições discretas **x** do volume, o cálculo é feito a partir da cor e opacidade dos seus voxels vizinhos.

Neste trabalho é utilizada uma notação paramétrica para representar o raio. Sendo assim, s_i denota o i-ésimo ponto de amostragem usado para navegação ao longo de $\mathcal{R}_p$, que vai do pixel $I(\mathbf{u})$ em direção ao volume de dados, com $i \in [0, n]$, e sendo $n+1$ o número total de amostras. A obtenção do valor das amostras processadas $V(\mathbf{x}_{s_i})$ pode requerer uma interpolação do volume de dados discreto caso a amostra não corresponda a uma posição discreta **x** (por exemplo interpolação trilinear [22]).

As propriedades físicas de cada ponto amostrado s_i ao longo do raio $\mathcal{R}_p$ são consideradas para calcular a intensidade do pixel $I(\mathbf{u})$, de tal maneira que:

$$I(\mathbf{u}) = g(\mathcal{R}_p) \tag{1}$$

onde g é uma função que determina a cor de um pixel em função das amostras processadas ao longo do raio $\mathcal{R}_p$ que intercepta o volume V. Usualmente, a função g é implementada da seguinte maneira:

$$g(\mathcal{R}_p) = \sum_{s_i=a}^{b} f(\mathbf{x}_{s_i}), \tag{2}$$

onde $s_0 = a$ é a primeira amostra (frontal) que contribui para $I(\mathbf{u})$, $s_n = b$ é a última amostra (traseira), e $f(\mathbf{x}_{s_i})$ é uma função que determina a contribuição de cada ponto amostrado na intensidade final do pixel, sendo que cada um é processado de acordo com as propriedades visuais do voxel $V(\mathbf{x}_{s_i})$.

Na figura 1 e na equação 2 se considerou que o processamento do algoritmo de *ray casting* é realizado na ordem *front-to-back*: as amostras são processadas a partir do primeiro ponto de intersecção a do raio com o volume até o segundo ponto de intersecção b. Entretanto, também é possível realizar o processamento na ordem *back-to-front*, de b para a.

Tanto a função g como a função f podem ser implementadas de várias maneiras e com diferentes argumentos. Na abordagem RGBα introduzida por Levoy [20], f é uma função que faz o mapeamento dos valores dos voxels em cor e opacidade, que são, então, combinados com um modelo de iluminação simples. A formulação apresentada por Levoy, com processamento *back-to-front*, é rescrita a seguir de acordo com a notação adotada neste trabalho e considerando o processamento do tipo *post-shaded volume rendering*:

$$g(\mathcal{R}_p) = \sum_{i=0}^{K} [T_{C_\lambda}(\mathbf{x}_{s_i}) T_\alpha(\mathbf{x}_{s_i}) \prod_{j=i+1}^{K} (1 - T_\alpha(\mathbf{x}_{s_j}))] \tag{3}$$

onde: K é o número total de amostras processadas, sendo $i = 0$ a amostra mais distante do observador (*back-to-front*); $T_{C_\lambda}(\mathbf{x}_{s_i})$ representa a função de transferência que determina a cor da amostra s_i de acordo com $V(\mathbf{x}_{s_i})$ para uma determinada banda espectral, no caso RGB; $T_\alpha(\mathbf{x}_{s_i})$ representa a função de transparência que determina o valor de opacidade associado à amostra s_i. T_{C_λ} é a cor de fundo e T_α é 1 quando $i = 0$. Na prática, este processamento é feito para R, G e B separadamente.

2.2 Algoritmo de *Splatting*

Splatting [40] é um algoritmo de visualização do tipo *image-order* usado com volumes de dados regulares mas que podem não possuir o mesmo espaçamento nas três dimensões da grade. Este algoritmo tonaliza as amostras de entrada e reconstrói a imagem final a partir do volume tonalizado, ou seja, classificado e iluminado. Tomando planos paralelos ao plano da imagem, a reconstrução é realizada para todas as amostras em cada um desses planos. O conteúdo de cada plano é combinado posteriormente para formar a imagem final.

Este algoritmo é composto por quatro etapas principais: transformação, tonalização, reconstrução e visibilidade. Inicialmente, a amostra é processada através da sua transformação do espaço do volume $[x, y, z]$ para o espaço da tela $[u, v, w]$. Depois é feita a tonalização da amostra, que engloba classificação e iluminação, usando apenas a informação local. A amostra tonalizada consiste em uma tupla $[u, v, w, R, G, B, \alpha]$.

A reconstrução é realizada para todas as amostras em um plano do volume de dados, definido como um plano paralelo ao plano da imagem. Após a determinação da porção da imagem que a amostra influencia, através da função *footprint*, a sua contribuição é adicionada em um acumulador do plano. Quando todas as amostras de um plano do volume de dados são processadas, o conteúdo do acumulador do plano é combinado na imagem através de um operador de composição. Depois que todas as amostras forem processadas, é, então, obtida a imagem final.

A função *footprint* é calculada para cada posição de observação do volume de dados. Como cada voxel projetado (*splat*) é representado por um *kernel* de interpolação simétrico, primeiro é calculada a extensão do espaço da tela onde o *kernel* será projetado. Todos os pixels cujos centros se encontram na região onde o *kernel* é projetado podem ser afetados pela amostra. A partir do kernel, são ponderados os valores do voxels. A tabela que representa a função *footprint* é determinada em uma etapa de pré-processamento [40].

A grande vantagem deste algoritmo é que somente os voxels relevantes para a imagem devem ser projetados. Desta maneira, o volume de dados que precisa ser processado e armazenado é reduzido. Entretanto, dependendo do fator de *zoom*, cada *splat* pode englobar uma grande quantidade de pixels que precisam ser processados [26].

2.3 Algoritmo de *Shear-Warp*

O algoritmo do tipo *object-order* proposto por Lacroute e Levoy [19], chamado *shear-warp*, baseia-se na fatoração da matriz de visualização. Inicialmente o volume é transformado para um sistema de coordenadas intermediário (*sheared object space*), através de um mapeamento simples a partir do sistema de coordenadas do objeto. Neste novo sistema de coordenadas, todas as fatias do volume estão paralelas ao plano da imagem e perpendiculares aos raios (figura 2). A projeção resultante forma uma imagem intermediária distorcida, que é, então, corrigida (*warping*) para gerar a imagem final. A equação 4 descreve a fatoração da matriz de visualização:

$$M_{view} = PSM_{warp} \tag{4}$$

onde P é matriz que faz com que o eixo z seja paralelo aos raios, S transforma o volume para o *sheared object space*, e M_{warp} faz a transformação para coordenadas da imagem.

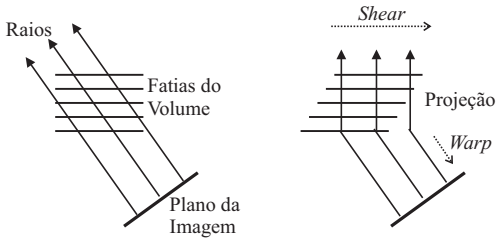

Figura 2. Fatoração de *shear-warp* para projeção paralela [19]

Resumidamente, então, o algoritmo é composto das seguintes etapas: (i) transformação do volume de dados para o *sheared object space*, transladando e reamostrando cada fatia de acordo com S; (ii) composição das fatias reamostradas, projetando o volume numa imagem 2D intermediária; (iii) transformação da imagem intermediária para o espaço da imagem final, corrigindo a deformação de acordo com M_{warp}.

A projeção no *sheared object space* possui várias propriedades geométricas que simplificam a etapa de composição do algoritmo (ii). Primeiro, os pixels a serem percorridos na imagem intermediária são paralelos aos voxels no volume de dados, e a todos os voxels é aplicado o mesmo fator de escala. Na projeção paralela, cada *voxel slice* possui o mesmo fator de escala, que pode ser escolhido arbitrariamente. Neste caso, é possível escolher um fator de escala unitário de tal maneira que exista um mapeamento um-para-um entre os voxels e os pixels da imagem intermediária.

Considerando a primeira destas propriedades, observa-se que as estruturas de dados da imagem e do volume podem ser percorridas na mesma ordem, linha a linha. Portanto,

Lacroute e Levoy [19] usaram o método de compactação *Run-Lenght Encoding* para os voxels, de tal forma que o algoritmo não processa os voxels transparentes. A codificação consiste em dois tipos de *runs*, transparente e opaco, definidos através de um *threshold* especificado pelo usuário. Para aproveitar a coerência da imagem e do volume, um *offset* para o próximo pixel não opaco na mesma linha é armazenado com cada pixel opaco da imagem intermediária, de tal forma que conjuntos de pixels opacos não precisam ser examinados. Esta compactação da imagem intermediária, que é equivalente à terminação adaptativa do raio no algoritmo de *ray casting* [21], é processada durante o *rendering*. Desta maneira, são processados apenas os voxels visíveis que não são transparentes, evitando o processamento de voxels sobre pixels opacos e, consequentemente, diminuindo o tempo de geração das imagens, principalmente em relação ao algoritmo de *ray casting*.

Quando os voxels são processados, é realizada a coloração, amostragem e composição, sendo que a etapa de amostragem também pode ser simplificada. Considerando a segunda e a terceira propriedades geométricas, como a transformação que é aplicada a cada fatia do volume antes da projeção consiste apenas em uma translação, os pesos para a amostragem são os mesmos para cada voxel na mesma fatia e não precisam ser calculados separadamente. Para processar uma linha da imagem duas linhas de voxels são acessadas simultaneamente e um filtro de interpolação bilinear é utilizado.

2.4 Visualização de Estruturas Internas

Muitos trabalhos descritos na literatura apresentam maneiras de incorporar aos algoritmos de visualização direta de volume a capacidade de visualizar estruturas internas, ou em outras palavras, de "ver através" do volume. Dois tipos principais de abordagens são identificados nas técnicas existentes: remoção dos voxels que estão fora da região de interesse e controle da transparência do voxel durante a etapa de classificação. A primeira abordagem, **remoção dos voxels**, pode ser tratada como complementar à tarefa de seleção, onde são selecionados apenas os voxels que estão dentro da região de interesse. A tarefa de remover (ou selecionar) voxels pode ser feita, basicamente, através de duas técnicas diferentes: segmentação e corte [43].

As técnicas de segmentação possuem como objetivo isolar um ou vários objetos de interesse, de maneira que eles possam ser exibidos como modelos geométricos. Por outro lado, a segmentação também pode ser usada para isolar e remover uma determinada estrutura do volume. Por exemplo, técnicas de segmentação manual podem ser usadas para extração de contornos em cada fatia de um volume de dados, com o objetivo de identificar os voxels que correspondem à ROI. A partir destes contornos, usando uma malha de triângulos, é gerada a superfície que deve ser visualizada [38]. No método descrito por Bullitt e Aylward [3] é feita uma combinação do algoritmo de *ray casting* com objetos tubulares segmentados, tais como vasos sangüíneos, que constituem as estruturas de interesse. Neste caso, após

a etapa de segmentação, são processados apenas os raios que interseccionam o objeto. Já no trabalho apresentado por Kanitsar et al. [10], uma ferramenta de segmentação interativa é usada para remover ossos de imagens de angiografia por tomografia computadorizada [3]. Neste caso, usando uma composição do tipo MIP[4], toda a estrutura dos vasos sangüíneos pode ser visualizada.

Técnicas de corte (seção 3) preocupam-se com a remoção de partes do volume que o usuário não deseja ver, revelando a ROI e fornecendo uma melhor compreensão do conjunto de dados. Por exemplo, um plano de corte pode ser usado para expor uma nova superfície e permitir a visualização dos objetos do seu interior. Inicialmente, este conceito foi introduzido como *volume slicing*, que consiste em visualizar o volume de dados como imagens 2D ortogonais aos três eixos principais ou paralelas ao plano de visualização. Posteriormente houve uma evolução para ferramentas de corte mais flexíveis, que podem ser usadas para remover (ou selecionar) blocos de voxels de várias maneiras.

Na segunda abordagem, **controle da transparência do voxel**, a idéia geral é manipular a visibilidade das estruturas através do controle das amostras que serão consideradas ao longo do raio, de forma que as estruturas relevantes para um determinado estudo sejam destacadas (seção 4). No trabalho pioneiro de Levoy [20], e em vários outros algoritmos de visualização direta de volume, tabelas de classificação são usadas para atribuir cor e opacidade a diferentes intervalos de valores de voxels. Tais tabelas representam as funções de transferência que fazem o mapeamento dos valores dos voxels para propriedades visuais. Assim, um valor de opacidade é especificado para cada estrutura identificada. Dependendo da função de transferência utilizada, se a transparência também estiver associada à magnitude do vetor gradiente [22], as superfícies existentes no volume são realçadas, de tal maneira que as estruturas são facilmente distinguidas. A opacidade também pode estar associada à posição do voxel no volume, ou à profundidade da amostra ao longo do raio, como descrevem Mullick et al. [28], no algoritmo denominado *Confocal Volume Rendering*.

3 O Uso de Ferrametas de Corte para Inspeção de Volumes

Cortar (ou recortar) significa eliminar do processamento pixels ou primitivas gráficas que não estejam na janela de seleção ou região de interesse. A seleção de uma seção oblíqua, e a utilização de um plano de corte para eliminar uma parte dos voxels do processamento, são alguns exemplos de ferramentas de corte utilizadas na área de visualização volumétrica. Na literatura, estas ferramentas tipicamente correspondem a elementos geométricos usados

[3]Angiografia consiste na imagem dos vasos sangüíneos obtida pela injeção de material de contraste na corrente sangüínea.
[4]MIP (*Maximum Intensity Projection*) é uma operação de composição que consiste em escolher o maior valor escalar encontrado ao longo de um raio [22].

para definir o subconjunto de voxels que será removido. De maneira complementar, usando estas mesmas ferramentas, é possível selecionar apenas o subconjunto de voxels de interesse para o processamento. Além disso, propriedades dos voxels também podem ser usadas para definir qual é a parte do volume de dados que será recortada ou selecionada.

Assim, do ponto de vista do usuário, tais ferramentas podem ser usadas para controlar a região de interesse de duas maneiras diferentes: **por exclusão**, quando as ferramentas são usadas para especificar toda a região do volume que será retirada do processamento; **por inclusão**, quando as ferramentas são usadas para especificar toda a região do volume onde será feito o processamento. Por exemplo, no caso de um volume de corte, ou as amostras são processadas nas posições do raio que estão fora do volume, ou são processadas somente nas posições do raio que estão no seu interior. Neste trabalho se assume que a operação de corte é equivalente ao processamento por exclusão, e a operação de seleção, ao contrário, é equivalente ao processamento por inclusão, pois somente as amostras da ROI são processadas (de acordo com o exemplo, o interior do volume de corte).

Considerando o algoritmo de *ray casting*, na seção 3.1, é introduzida uma abordagem unificada para as técnicas de corte, por exclusão e por inclusão, que permitem visualizar o interior dos volumes de dados. As seções 3.2 e 3.3 apresentam diversas técnicas de corte.

3.1 Ferramentas de Corte na Visualização Direta de Volume

Conforme o esquema geral apresentado na seção 2.1, um raio $\mathcal{R}_p$ é lançado para cada pixel $I(\mathbf{u})$, e sua intensidade é determinada pelo processamento das amostras s_i ao longo do raio (equação 2). Considerando a ordem de processamento *front-to-back*, o intervalo de amostragem ao longo de $\mathcal{R}_p$ é dado por [a,b]. Na ausência de ferramentas de corte, a e b correspondem, respectivamente, aos dois pontos de intersecção de cada raio com o volume de dados (figura 3): r_1, que indica a posição onde o raio entra no volume de dados, e r_2, que corresponde à posição onde o raio sai do volume (deste modo, $a = r_1$ e $b = r_2$).

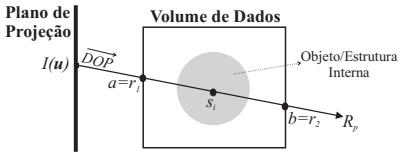

Figura 3. Pontos de intersecção (r_1,r_2) do raio com o volume de dados (ver figura 1)

As ferramentas de corte podem ser tratadas como uma estratégia para modificar o intervalo de amostragem através da especificação de a e b para valores diferentes dos pontos de intersecção do raio com o volume de dados. A figura 4 ilustra a utilização de um plano de

corte frontal e um traseiro, cuja intersecção com o raio é dada por c_1 e c_2. Assim, o intervalo de amostragem é determinado por $a = c_1$ e $b = c_2$.

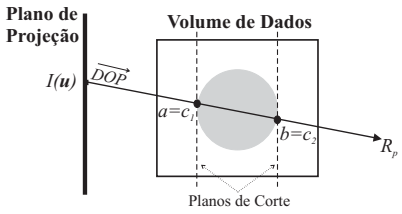

Figura 4. Planos de corte (c_1,c_2) delimitam o intervalo de amostragem em um raio

Seguindo esta linha de raciocínio, as ferramentas de corte podem ser generalizadas como duas funções, f_a e f_b, que são usadas para determinar os valores de a e b:

$$a = f_a\big(U_a, \mathcal{R}_p, V\big)\ b = f_b\big(U_b, \mathcal{R}_p, V\big) \tag{5}$$

onde U_a e U_b correspondem a parâmetros fornecidos interativamente pelo usuário para configurar a ferramenta de corte, $\mathcal{R}_p$ é o raio e V é o volume de dados. Diferentes ferramentas de corte podem ser implementadas dependendo da escolha dos parâmetros U e das funções f_a e f_b.

Os parâmetros U tipicamente correspondem a elementos geométricos, tais como um plano ou volume de corte, usados para definir o subconjunto de voxels que deve ser removido. Neste caso, a função f processa o ponto de intersecção entre o raio e o elemento geométrico para determinar a posição de a e b para um dado pixel. Se não há intersecção, assume-se que $f_a = r_1$ e $f_b = r_2$. Entretanto, em algumas técnicas, os parâmetros U podem ser um intervalo de valores de densidade de voxel que pertencem a uma região de interesse. Considerando este esquema genérico, é feita uma discussão de como as ferramentas de corte mais utilizadas descritas na literatura implementam U_a, U_b, f_a e f_b: se consideram apenas os elementos geométricos (seção 3.2) ou se consideram também o conteúdo do volume de dados (seção 3.3).

Para exemplificar as ferramentas de corte, bem como as funções de transferência descritas na seção 4, foram utilizados dois volumes de dados sintetizados e um volume de dados de MRI de uma cabeça (figura 5), o qual foi adquirido por ftp da Universidade de Carolina do Norte (EUA). Os volumes sintetizados consistem em esferas que estão ilustradas nas figuras 5a e 5b. A esfera da figura 5a possui três camadas com densidades diferentes e um quadrado no seu interior. A esfera da figura 5b possui apenas uma camada externa e uma série de paralelepípedos no seu interior. Para analisar as ferramentas de corte foram sempre utilizados os mesmos parâmetros de visualização e uma função de transferência de opacidade linear (ver seção 4).

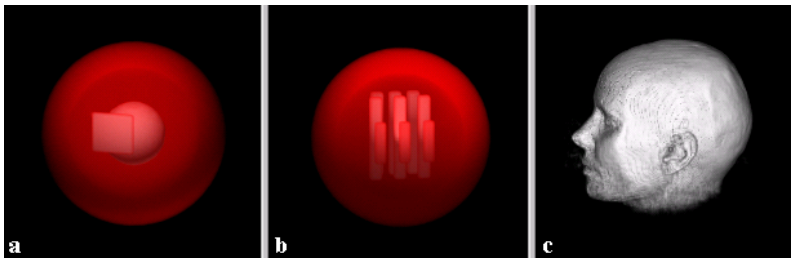

Figura 5. Volumes usados para ilustrar a descrição das técnicas de corte e funções de transferência

3.2 Corte por Geometria

As ferramentas para extração de seções, também conhecidas como *slicing tools*, permitem a visualização de um corte 2D do volume (espessura = 1 voxel). Neste caso, como $a = b$, somente uma amostra s_i é usada no processamento de cada pixel $I(\mathbf{u})$. Assim, $f = f_a = f_b$ determina a localização da amostra através do cálculo de intersecção entre $\mathcal{R}_p$ e um plano. Este plano corresponde à informação de geometria que é fornecida pelo usuário através da configuração de $U = U_a = U_b$.

Para métodos de corte com **seções planares**, ortogonais ou oblíquas, U consiste na definição da posição e orientação de um plano de corte. No caso de uma seção ortogonal, o plano é sempre ortogonal a um dos três eixos principais do sistema de referência do objeto, que neste caso é o volume de dados. Para uma seção oblíqua o plano é orientado arbitrariamente. Considerando o corte com **seções não-planares**, como por exemplo uma seção curvilínea [34], também é necessário processar o ponto de intersecção entre $\mathcal{R}_p$ e, neste caso, uma superfície. Esta superfície, especificada através de U, pode ser uma malha ou uma superfície paramétrica.

Sendo assim, conclui-se que a geração de seções ortogonais, oblíquas e não-planares consiste em um corte por inclusão, uma vez que o usuário sempre especifica o plano ou superfície onde será feita a amostragem. Os cortes coronal, sagital e axial ilustrados na figura 6 são exemplos de seções ortogonais. No exemplo de uma seção oblíqua na figura 6, o observador está posicionado a frente e acima do volume.

Um conjunto de ferramentas mais flexível, onde $a \neq b$, envolve o processamento de várias amostras para cada $\mathcal{R}_p$, resultando em um sub-volume. Neste caso, planos de corte adicionais são introduzidos no espaço do objeto (volume de dados) [22]. Uma primeira possibilidade consiste em especificar um **plano de corte frontal** para remover todos os voxels que estão entre o plano de projeção e este plano [41]. Assim, U_a consiste na definição dos parâmetros do plano, de tal forma que $f_a = c_1$ e $f_b = r_2$. De maneira análoga, o usuário

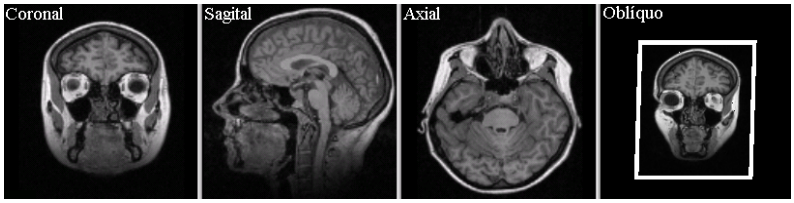

Figura 6. Exemplos de planos de corte ortogonais e oblíquo

também pode especificar **um plano de corte traseiro** através da configuração dos parâmetros do plano dada por U_b. Desta maneira $f_b = c_2$, isto é, todos os voxels localizados após este plano são também removidos. Esta classe de ferramenta de corte pode ser exemplificada pelo trabalho de Yen et al. [43], que apresentaram um método onde o *rendering* é feito entre um conjunto de planos de corte paralelos, perpendiculares à direção de visualização.

A utilização de planos de corte para a especificação de regiões onde o processamento será realizado (corte por inclusão) ou não (corte por exclusão) é exemplificada nas figuras 7a, 7b e 7c. A figura 7a mostra a utilização de um plano de corte frontal ($f_a = c_1$ e $f_b = r_2$). Na figura 7b foram utilizados dois planos de corte para definir a ROI, de tal maneira que $f_a = c_1$ e $f_b = c_2$. A figura 7c é um exemplo da utilização de planos de corte como ferramentas de corte por inclusão, uma vez que são processadas apenas as amostras que estão exatamente sobre os planos. Desta maneira, diferentes planos que pertencem ao volume de dados podem ser exibidos, permitindo a visualização do interior do objeto.

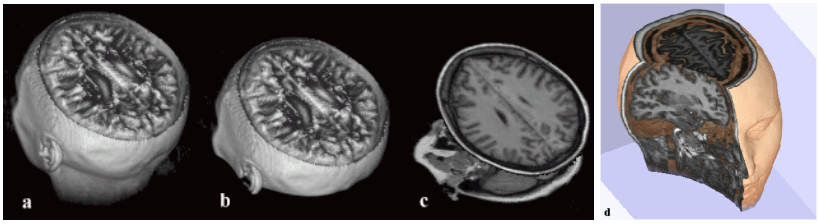

Figura 7. Exemplo da utilização de um (a) ou dois planos de corte (b, c) e de uma superfície de corte não-planar [35] (d)

Uma alternativa para a utilização de planos de corte, é o uso de **superfícies de corte não-planares**, exemplificado por Silva [35] (figura 7d). A diferença em relação ao plano de corte está nos parâmetros U_a e U_b, que neste caso especificam objetos 3D mais complexos, e nas funções que calculam os pontos de intersecção c_1 e c_2, que também são menos triviais do que no caso do cálculo de intersecção de um plano com uma reta (raio).

Um **objeto volumétrico convexo de corte** também pode ser usado para eliminar todos os voxels localizados na intersecção do mesmo com o volume, isto é, que estão **dentro** do seu interior [35] (corte por exclusão). Este objeto pode ser um cubo, uma esfera ou pode ter uma geometria qualquer. No exemplo da figura 8a, U_a determina um paralelepípedo de corte. Neste caso $f_b = r_2$ e a função f_a processa a intersecção entre este paralelepípedo e $\mathcal{R}_p$. Se não houver intersecção, $f_a = r_1$; se houver intersecção, $f_a = c_2$, que corresponde à posição na qual o raio deixa o paralelepípedo de corte. No caso de um volume de corte estar posicionado no meio do volume de dados, um processamento especial é necessário para gerenciar dois intervalos de amostragem: $f_a = r_1$ e $f_b = c_1$, que corresponde às amostras que estão mais próximas do plano de projeção; $f_a = c_2$ e $f_b = r_2$, que correspondem às amostras que estão posicionadas depois do objeto de corte ao longo do raio (mais afastadas do plano de projeção).

De maneira complementar, um objeto volumétrico pode ser usado para especificar o volume de interesse (ou VOI) através do corte de todos os voxels que estão **fora** do seu interior. Neste caso de corte por inclusão, somente as amostras dentro do objeto geométrico são consideradas no cálculo da cor final do pixel. As funções f_a e f_b eqüivalem, respectivamente, ao primeiro e segundo ponto de intersecção entre o volume de corte dado por $U_a = U_b$ e $\mathcal{R}_p$, de tal maneira que $f_a = c_1$ e $f_b = c_2$.

Um experimento usando um paralelepípedo como volume de corte é exibido na figura 8a. Para gerar esta imagem, primeiro foram definidas as dimensões e a localização deste volume de corte. Depois, para cada raio, os pontos de intersecção são processados e o parâmetro a corresponde ao ponto de saída do raio do volume de corte. Para a seleção do sub-volume (figura 8b), o paralelepípedo é usado para selecionar a ROI, o que faz com que todos os voxels que estão fora do seu interior sejam eliminados e, consequentemente, são exibidos apenas aqueles que estão no seu interior.

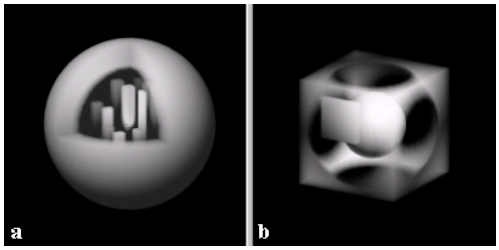

Figura 8. Exemplo da utilização de um paralelepípedo para realizar o corte por exclusão (a) e por inclusão (b)

Além de primitivas gráficas, tais como cubo e esfera, outros objetos volumétricos também podem ser usados para realizar o corte, sendo necessário apenas alterar o parâmetro

U e o processamento dos pontos de intersecção. No trabalho de Westermann e Ertl [41], por exemplo, a visualização direta de volume é baseada em textura 3D e permite a extração de um sub-volume ou VOI através do uso de uma geometria arbitrária para o corte interativo.

3.3 Corte por Geometria e Conteúdo

Nas técnicas de corte descritas, U_a e U_b consistem em informações geométricas, tal como um plano de corte, por exemplo. A técnica de *Confocal Volume Rendering* (CVR) apresentada por Mullick et al. [28], entretanto, introduz uma noção diferente, onde as propriedades do voxel são usadas em conjunto com a informação geométrica para especificar a e b. CVR é usado para controlar o "foco" numa região através de três parâmetros:

1. S-Band (*See-Through Band*): usado para indicar a região onde não haverá processamento; consiste na definição de uma propriedade do voxel (tal como um intervalo de densidade) que caracteriza a superfície externa de uma estrutura, e de uma distância (por exemplo em mm) a partir desta superfície;
2. T-Band (*Transition Band*): indica exatamente o quanto da estrutura em termos de profundidade (por exemplo, em mm) o usuário deseja visualizar a partir do final da S-Band;
3. Função de Escala (*Depth Enhance*): configura a função de escala de opacidade dentro da porção visível do volume, permitindo que o usuário realce ou identifique como uma estrutura específica dentro da ROI será exibida.

O primeiro parâmetro (S-Band) corresponde a U_a, que é usado para definir a de acordo com a propriedade do voxel $P(V(\mathbf{x}_{s_i}))$ e uma distância constante d_1. Considerando que v é o valor procurado que representa a estrutura de interesse, no caso a superfície mais externa de um objeto, conclui-se que: $P(V(\mathbf{x}_{s_i}))$ é verdadeiro se $V(\mathbf{x}_{s_i}) = v$, e falso se $V(\mathbf{x}_{s_i}) \neq v$. Assim:

$$f_a = first(\mathcal{R}_p, v) + d_1 \tag{6}$$

onde $first$ é a função que denota a posição s_i procurada, ou seja, a primeira amostra s_i no raio $\mathcal{R}_p$ na qual $P(V(\mathbf{x}_{s_i})) = Verdadeiro$.

O segundo parâmetro, T-Band, corresponde a U_b, que é usado para especificar b de acordo com a e uma distância constante d_2 da seguinte maneira:

$$f_b = a + d_2 \tag{7}$$

Observa-se, então, que CVR corresponde a uma técnica de corte por inclusão, sendo o intervalo de amostragem $[a, b]$ determinado por uma combinação das propriedades do voxel (em

f_a) e da geometria (distâncias em f_a e f_b). A figura 9 mostra exemplos da utilização desta técnica. As linhas I e II que aparecem nas vistas axial e sagital correspondem aos parâmetros S-Band e T-Band, respectivamente.

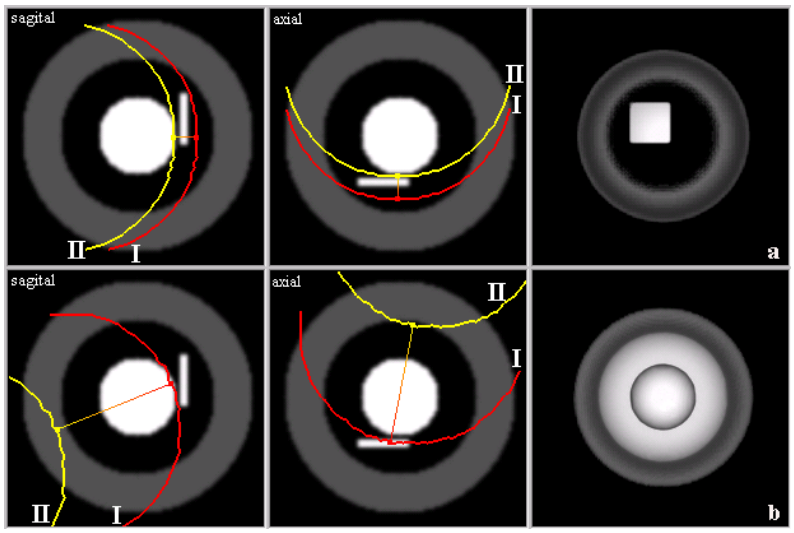

Figura 9. Técnica de *Confocal volume rendering*

Uma técnica usada para exibir informações vasculares, apresentada por Zuiderveld [44], também pode ser considerada uma técnica de corte. *Closest Vessel Projection* (CVP) consiste em terminar de processar as amostras quando o raio deixa o primeiro vaso sangüíneo encontrado. Esta técnica, que é usada como uma alternativa para a composição do tipo MIP, tem como objetivo aumentar a percepção de profundidade das imagens geradas. No CVP, f_a corresponde ao primeiro ponto de intersecção com o volume (r_1), e f_b é especificado de acordo com a propriedade do voxel $P(V(\mathbf{x}_{s_i}))$ para $\mathbf{s}_i > a$, que neste caso é a densidade do limiar do vaso sangüíneo. Assim, b pode ser processado da seguinte maneira:

$$f_b = second(\mathcal{R}_p, v) \tag{8}$$

onde $second$ é a função que denota a posição s_i procurada, ou seja, a segunda amostra s_i ao longo do raio $\mathcal{R}_p$ na qual $P(V(\mathbf{x}_{s_i})) = Verdadeiro$, considerando que v representa o valor da superfície externa do vaso sangüíneo. Se o valor de $P(V(\mathbf{x}_{s_i}))$ não for encontrado, $f_b = r_2$. Esta técnica também pode ser considerada como um corte por inclusão.

4 Funções de Transferência

A função de transferência é responsável pela associação de propriedades visuais, tais como cor e opacidade, aos valores originais do volume de dados que está sendo visualizado [30]. Estas funções, usadas na etapa de classificação do algoritmo de *ray casting*, permitem explorar as estruturas existentes nos volumes de dados sem a necessidade de definir explicitamente a sua forma ou extensão. Para alcançar este objetivo, geralmente, um valor de opacidade é associado a cada voxel para descrever quanto de energia luminosa é absorvido, indicando se é possível ou não enxergar através desse voxel, e permitindo a definição de estruturas transparentes, semitransparentes e opacas [22].

A tarefa de especificar funções de transferência que gerem imagens de qualidade e que transmitam as informações requeridas não é trivial, e tem sido amplamente discutida [30]. Um problema, por exemplo, é que normalmente os valores de opacidade são definidos através da edição de gráficos que possuem pontos de controle que podem ser alterados. O processo torna-se mais complexo ainda quando há interação com outros parâmetros, tais como fonte de luz e modelo de tonalização [15]. Desta maneira, encontrar uma função de transferência de cor ou opacidade tende a ser uma tarefa demorada e realizada através de "tentativa e erro". Por isso, recentemente houve um aumento na pesquisa de técnicas automáticas e semi-automáticas para a criação destas funções, e no desenvolvimento de interfaces interativas mais amigáveis para auxiliar nesta tarefa.

Inicialmente, assume-se a existência de uma função de transferência genérica T, usada para atribuir propriedades visuais a um voxel. Esta função pode considerar ou ter como argumento uma ou mais propriedades do voxel, conforme mostra a equação 9:

$$v_o = T\big(V(\mathbf{x}_{s_i}), \|\nabla V(\mathbf{x}_{s_i})\|, \nabla^2 V(\mathbf{x}_{s_i}), \dots\big) \tag{9}$$

onde v_o é a propriedade visual do voxel (cor ou opacidade), T é a função de transferência, $\|\nabla V(\mathbf{x}_{s_i})\|$ é a magnitude do gradiente no ponto de amostragem e $\nabla^2 V(\mathbf{x}_{s_i})$ é o Laplaciano, um operador da derivada de segunda ordem. Existem diversas implementações de T, e cada uma pode utilizar diferentes propriedades do voxel.

Do ponto de vista de visualização de estruturas internas, a função T que mais influencia o resultado final é a função de transferência de opacidade. A função para atribuir cor é necessária porque a maioria dos volumes de dados não possuem valores de cor associados a cada voxel [22], e a utilização de cores pode ser muito útil, principalmente para distinguir e caracterizar estruturas. As seções seção 4.1 e seção 4.2 apresentam algumas implementações de T já descritas na literatura, incluindo as técnicas e ferramentas utilizadas para especificação das funções.

4.1 Funções de Transferência de Opacidade

As funções de transferência de opacidade (T_α) mais simples especificam a opacidade apenas de acordo com a intensidade do voxel, ou de acordo com a intensidade do voxel e a magnitude do gradiente local. Sendo assim, neste trabalho é usada a seguinte expressão genérica para representar uma função de transferência de opacidade:

$$\alpha(\mathbf{x}_{s_i}) = T_\alpha\big(\mathbf{x}_{s_i}\big) \tag{10}$$

onde o valor de opacidade da amostra $\mathbf{x}_{s_i} \in [0,1]$ e T_α é a função de transferência que determina um valor de opacidade.

Atualmente existe um conjunto padrão de funções de transferência de opacidade que normalmente é fornecido e/ou utilizado nos sistemas. Entre estas funções destacam-se: rampa, trapézio, bloco, exponencial e linear, esta, a mais simples de todas. O mapeamento destas funções, que estão descritas a seguir, normalmente é realizado considerando apenas a intensidade do voxel.

Numa **função linear**, quanto maior for a intensidade do voxel, maior é o valor de opacidade que ele recebe. T_α neste caso é calculada através de $V(\mathbf{x}_{s_i})/I_m$, onde I_m é o maior valor de intensidade que um voxel do volume de dados pode ter. No caso de dados médicos, por exemplo, também é possível desprezar valores de intensidade muito pequenos, que geralmente correspondem a ruídos da imagem, atribuindo zero para a opacidade. Assim, o cálculo da função linear seria:

$$T_\alpha\big(\mathbf{x}_{s_i}\big) = \begin{cases} 0, & V(\mathbf{x}_{s_i}) < I_l \\ (V(\mathbf{x}_{s_i}) - I_l)/(I_m - I_l), & V(\mathbf{x}_{s_i}) \geq I_l \end{cases} \tag{11}$$

onde, I_l é o valor de intensidade que corresponde ao limiar (ou *threshold*), responsável pela identificação dos voxels que devem ser considerados no processamento. A figura 10 ilustra a diferença da aplicação de uma função de transferência linear sem desprezar valores de intensidade pequenos (figura 10a) e usando a 11 com a atribuição do valor 75 para I_l (figura 10b).

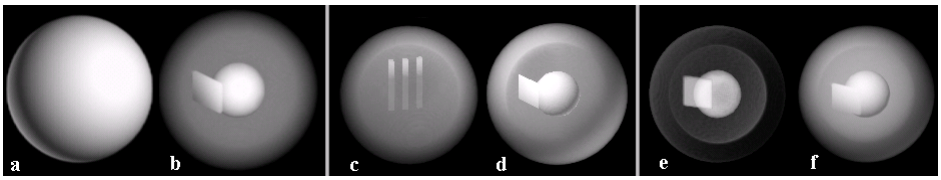

Figura 10. Exemplos da aplicação de funções de transferência linear

A função chamada de **"rampa"** por König e Gröller [18], é uma variação da função linear, que ocorre quando o valor de intensidade do voxel que possui opacidade 1 não é o maior, mas sim o central. Sua equação é:

$$T_\alpha(\mathbf{x}_{s_i}) = \begin{cases} \frac{V(\mathbf{x}_{s_i})}{I_c}, & V(\mathbf{x}_{s_i}) \leq I_c \\ \frac{(I_m - V(\mathbf{x}_{s_i}))}{I_c}, & V(\mathbf{x}_{s_i}) > I_c \end{cases} \tag{12}$$

onde I_c é o valor de intensidade médio (entre 0 e I_m) que um voxel pode ter.

König e Gröller [18] mostram uma forma muito utilizada para T_α, onde os valores são definidos de tal maneira que o formato do gráfico da função é um **trapézio**. Neste caso, os valores dos voxels centrais são mapeados para opacidade máxima, enquanto os voxels dos extremos são mapeados para valores crescentes ou decrescentes de opacidade, formando, assim, um trapézio.

Outro exemplo de função de transferência simples que também é muito utilizada é a **função exponencial**. Entretanto, quando há necessidade de atribuir um mesmo valor de opacidade a um conjunto de voxels cujas intensidades estão num determinado intervalo, se utiliza uma função de transferência do tipo **bloco**. Um exemplo de equação para esta função é apresentado na equação 13:

$$T_\alpha(\mathbf{x}_{s_i}) = \begin{cases} 1, & I_c \leq V(\mathbf{x}_{s_i}) \leq I_m \\ 0, & \text{Caso contrário} \end{cases} \tag{13}$$

É importante salientar que a equação 13 é apenas um exemplo, pois normalmente o mapeamento é realizado através da definição de mais do que um bloco, ou intervalo de intensidades. Neste caso, basta aumentar o número de alternativas da equação 13, acrescentando uma para cada bloco. Para ilustrar, as figuras 10c e 10d exemplificam o uso desta função onde a opacidade dos voxels de densidade entre 79 e 81 foi mapeada para o valor 0.05, entre 240 e 255 para o valor 1, e das demais densidades para o valor 0.

Considerando que a equação 13 pode ter um grande número de alternativas, uma forma mais simples de atribuir diferentes valores de opacidade para vários intervalos de intensidade dos voxels é através de uma **tabela**. Assim, por exemplo, um alto valor de opacidade pode ser mapeado para todos os voxels cujas intensidades estejam em um ou vários intervalos que identificam a(s) estrutura(s) de interesse, e um valor de opacidade baixo pode ser atribuído para outras estruturas que não são de interesse, de tal forma que fiquem transparentes. Com o uso de tabelas de classificação, geralmente criadas interativamente pelo usuário, cada valor de intensidade também pode ser mapeado para um valor de opacidade diferente. Esta tabela pode ser interpretada como um vetor cujo índice indica o valor de intensidade e o conteúdo o valor de opacidade [35]:

$$T_\alpha(\mathbf{x}_{s_i}) = vet[round(V(\mathbf{x}_{s_i}))] \tag{14}$$

onde, vet é o vetor representando a tabela e $round$ é uma função de arredondamento. Os valores de opacidade podem ser atribuídos de diversas maneiras nesta tabela. Uma possibilidade, consiste em inserir um ou mais pontos de controle e especificar para cada um deles o valor de opacidade. As demais posições da tabela, entre os pontos, são preenchidas através da utilização de uma função de interpolação, tal como a função linear.

Até aqui foram apresentadas as funções de opacidade mais simples, utilizadas quando a atribuição de um valor de opacidade para um voxel apenas de acordo com a sua intensidade é suficiente, como no caso dos volumes de dados de CT. Entretanto, para outros volumes, como por exemplo MRI, voxels que fazem parte do tecido cérebral e da pele podem ter a mesma intensidade. Assim, torna-se inviável fazer com que apenas uma destas estruturas, tal como a pele, seja transparente ou semitransparente para mostrar o interior do volume. Neste caso, é necessário usar outras informações para definir a função de transferência. O mais usual é usar a **magnitude do gradiente** local. Quanto maior for este valor, significa que há uma rápida mudança nos dados, indicando que existe uma superfície ou uma borda entre dois materiais [22].

Levoy [20] usou a intensidade do voxel e a magnitude do gradiente na etapa de classificação para a extração de uma isosuperfície. Naquele trabalho, era atribuída uma opacidade α_v para voxels que possuíam um valor selecionado I_v, e uma opacidade 0 para todos os outros voxels. Entretanto, para evitar o *aliasing*, voxels com valores próximos de I_v deveriam ter uma opacidade próxima de α_v. A aproximação adotada foi tal que a opacidade diminuia conforme o valor de intensidade se afastava de I_v, numa razão inversamente proporcional à magnitude do gradiente. Este mapeamento foi feito usando a equação 15:

$$T_\alpha(\mathbf{x}_{s_i}) = \begin{cases} 1, & \text{Se } \|\nabla V(\mathbf{x}_{s_i})\| = 0 \text{ e } V(\mathbf{x}_{s_i}) = I_v \\ 1 - \frac{1}{r}\left|\frac{I_v - V(\mathbf{x}_{s_i})}{\|\nabla V(\mathbf{x}_{s_i})\|}\right|, & \text{Se } \|\nabla V(\mathbf{x}_{s_i})\| > 0 \text{ e} \\ & V(\mathbf{x}_{s_i}) - r\|\nabla V(\mathbf{x}_{s_i})\| \leq I_v \leq V(\mathbf{x}_{s_i}) + r\|\nabla V(\mathbf{x}_{s_i})\| \\ 0, & \text{Caso contrário} \end{cases} \quad (15)$$

onde, r é a espessura desejada para a região de transição em voxels. A figura 11 ilustra a utilização desta função de transferência com a atribuição de diferentes valores para as variáveis I_v e r: 225 e 2.0, respectivamente, para geração da figura 11a; e 255 e 0.6 para a figura 11b.

Para ilustrar a diferença do uso da magnitude do gradiente como argumento na função de transferência de opacidade, as figuras 10e e 10f apresentam duas imagens geradas a partir da utilização de duas fórmulas diferentes. Na figura 10f, foi utilizada uma função linear onde os valores de intensidade menores do que 70 são desprezados. A figura 10e é resultado da utilização de uma função de transferência baseada na magnitude do gradiente, calculada de

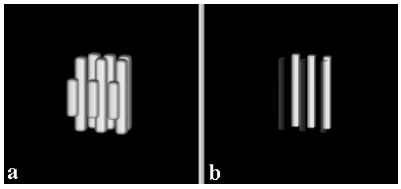

Figura 11. Utilização da função de transferência de opacidade apresentada por Levoy [20]

acordo com a equação 16, com $I = 40$.

$$T_\alpha(\mathbf{x}_{s_i}) = \begin{cases} 0, & V(\mathbf{x}_{s_i}) < I \\ \frac{\|\nabla V(\mathbf{x}_{s_i})\|}{\|\nabla V\|_m}, & V(\mathbf{x}_{s_i}) \geq I \end{cases} \tag{16}$$

Outros parâmetros foram recentemente introduzidos para definir a função de transferência: a **segunda derivada** [15] e o **histograma** da imagem [23]. Enquanto a magnitude do gradiente é uma aproximação da derivada de primeira ordem, um operador utilizado para aproximar a derivada de segunda ordem é o Laplaciano ($\nabla^2 V$) [8]. Ambos os operadores são utilizados para detecção de arestas. Em outras palavras, a máxima local da magnitude do gradiente identifica uma aresta (ou superfície), da mesma maneira que o valor mínimo da segunda derivada. Portanto, uma alternativa para detecção de arestas ou isosuperfícies é localizar o zero da segunda derivada ou usar estas duas informações em conjunto.

O histograma de uma imagem digital mostra uma estimativa da probabilidade de ocorrência de cada tonalidade [8]. No caso de um volume de dados, o histograma ($h(V)$) mostra quantos voxels existem para cada valor de intensidade. Portanto, através de um histograma se obtém informações que podem ajudar na escolha da função de transferência. Um exemplo é a facilidade para a identificação e posterior filtragem do ruído de uma imagem, bastando, neste caso, atribuir opacidade zero para todos os voxels com um determinado valor [22].

Liu et al. [23] trabalharam com a geração das funções de transferências a partir da análise do histograma de intensidades, que permite distinguir os diferentes materiais. Já Kindlmann e Durkin [15] desenvolveram um método onde um "histograma de volume" é criado através da utilização das seguintes propriedades: valor escalar (ou intensidade do voxel); primeira derivada aproximada pela magnitude do gradiente ($\|\nabla V\|$); e segunda derivada aproximada pelo Laplaciano ($\nabla^2 V$). Estas propriedades são na verdade os três parâmetros utilizados para a geração semi-automática de funções de transferência usadas para visualização de isosuperfícies. A partir do estudo do histograma, é possível extrair características das superfícies que representam a transição entre materiais de valores relativamente constantes, que sempre aparecem como uma curva de uma forma específica no histograma. O usuário

controla que porções das superfícies devem ser opacas sem a necessidade de saber os valores dos voxels.

4.2 Funções de Transferência de Cor

Além das funções de transferência de opacidade, o desenvolvimento de funções de transferência de cor também é muito importante, pois, em geral, não há uma cor associada a cada voxel dos volumes de dados, há apenas um valor de intensidade. A atribuição de cores é muito semelhante à etapa de classificação, e necessita de funções de transferência para transformar o valor de intensidade do voxel em uma cor RGB. Apesar do modelo RGB ser um dos mais utilizados nos algoritmos de visualização direta de volume, outros modelos de cor, tais como HLS (*Hue-Lightness-Saturation*) e HSV (*Hue-Saturation-Value*), também são utilizados.

Uma formulação genérica para função de transferência de cor, considerando o modelo RGB, seria:

$$R = T_{c_r}\left(\mathbf{x}_{s_i}\right) \tag{17}$$

$$G = T_{c_g}\left(\mathbf{x}_{s_i}\right) \tag{18}$$

$$B = T_{c_b}\left(\mathbf{x}_{s_i}\right) \tag{19}$$

onde T_{c_r}, T_{c_g} e T_{c_b} são as funções de transferência usadas para especificar os valores RGB, respectivamente. Estas funções podem ser implementadas da mesma maneira que algumas funções de opacidade, tais como linear, rampa e por tabela, sendo que neste caso, um valor RGB deve ser especificado para cada ponto de controle. Apesar da grande utilidade, é preciso ter muito cuidado na definição das tabelas de cores, pois da mesma maneira que o uso de cores pode melhorar a visualização, também pode prejudicar.

4.3 Abordagens para a Especificação de Funções de Transferência

Da análise dos vários trabalhos descritos na literatura é possível observar que as diferentes formas de especificar as funções de transferência variam, basicamente, de acordo com: os parâmetros utilizados (intensidade do voxel, magnitude do gradiente, segunda derivada, histograma); o grau de interação do usuário na especificação das funções (automática, semi-automática ou manual) e a abordagem adotada (tentativa e erro, centralizada na imagem, centralizada no dado, entre outras).

Neste trabalho, optamos por descrever as diferentes formas de especificação das funções de transferência seguindo a organização proposta por Kindlmann [16], ou seja, abordagens através de “tentativa e erro”; abordagens baseadas na detecção de características espaciais;

abordagens centralizadas na imagem; abordagens centralizadas no dado; e outras (baseadas em novos domínios e tipos de interação).

A especificação de funções de transferência através de tentativa e erro consiste em um método manual onde o usuário especifica as tabelas de cor e opacidade. Isto pode ser feito de diversas maneiras, mas o mais usual é permitir que pontos de controle, para os quais o usuário especifica a cor e opacidade, sejam inseridos e removidos das tabelas. Depois, uma função de interpolação é usada para preencher todas as posições intermediárias da tabela.

Segundo Kindlmann [16] as técnicas baseadas na detecção de características espaciais são extremamente flexíveis, mas não correspondem exatamente ao desenvolvimento de funções de transferências. Por exemplo, Fang et al. [6] apresentaram um modelo de função de transferência baseado em imagem que integra técnicas de processamento de imagem 3D, tais como realce e detecção de arestas, no *pipeline* de visualização. Nesta abordagem, as funções de transferência são representadas como uma seqüência de técnicas de processamento de imagens 3D, e permitem que os usuários ajustem um conjunto de parâmetros para gerar as imagens de acordo com o tipo de visualização desejada.

Nas abordagens centralizadas na imagem o foco está mais nas imagens geradas e na interface do sistema do que na especificação da função de transferência. Neste caso, onde a maioria dos sistemas utilizam funções simples, tais como linear e rampa, o usuário analisa as imagens e os parâmetros podem ser alterados até que seja gerado o resultado esperado. Assim, a geração das imagens depende da interação com o usuário, que seleciona imagens e/ou manipula parâmetros para a especificação da função de transferência para alcançar os resultados esperados. König e Gröller [18] introduziram um novo paradigma de interface para especificação de funções de transferência em sistemas de visualização de dados médicos. A idéia principal é que para cada intervalo de intensidades que contribuem para a imagem final, sua cor e sua opacidade são determinadas de forma independente, uma de cada vez, em interfaces separadas. O usuário seleciona o(s) intervalo(s) de interesse, seleciona uma função para fazer o mapeamento para valores de opacidade em cada intervalo (linear, rampa, trapézio, etc.) e, finalmente, seleciona uma função para especificar a cor também em cada intervalo.

Ao invés de analisar a imagem final, na abordagem centralizada nos dados, o usuário analisa as características do volume de dados para encontrar o isovalor que irá permitir a visualização da(s) superfície(s) de interesse. No trabalho desenvolvido por Bajaj et al. [1], *Contour Spectrum*, o objetivo é auxiliar o usuário a encontrar uma função de transferência adequada, identificando o melhor "isovalor" para extração das principais estruturas de um volume de dados. Esta técnica corresponde a uma assinatura que consiste no processamento em tempo real de propriedades de uma isosuperfície, tais como área da superfície de contorno e integral do gradiente. Estas propriedades são apresentadas para o usuário como uma coleção de gráficos que auxiliam na seleção de isovalores.

Outras abordagens de especificação de funções de tranferência não se enquadram em nenhuma das categorias caracterizadas acima. Novos domínios foram considerados no trabalho desenvolvido por Hladùvka et al. [9], que preocuparam-se com a definição semi-automática das funções de transferência no domínio das magnitudes da curvatura principal. As curvaturas principais consistem em dois números reais que indicam o quanto a normal varia nas direções dos vetores tangentes em cada ponto da superfície. Desta forma, as propriedades visuais são determinadas de acordo com a forma do objeto. Por outro lado, Kniss et al. [17] desenvolveram uma forma diferente de interação para definir funções de transferência multi-dimensionais, as quais se baseiam no valor de intensidade, na magnitude do gradiente e na segunda derivada [15]. Neste caso, um conjunto de *widgets* é manipulado para especificar as funções de transferência de forma intuitiva.

5 Comentários Finais

Com o avanço da tecnologia, tanto em poder de processamento como em precisão, tem aumentado consideravelmente o tamanho e a complexidade dos volumes de dados científicos a serem analisados por cientistas e profissionais das mais diversas áreas. Um exemplo são os dispositivos de aquisição de imagens médicas, que atualmente são capazes de gerar volumes de $512 \times 512 \times 512$ com o mesmo espaçamento entre pixels e entre fatias. Novas técnicas de processamento e visualização desses grandes volumes de dados são definitivamente necessárias de modo que a análise possa ser realizada em tempo hábil, por observadores humanos. Existe, ainda, dependendo da aplicação, a preocupação em gerar imagens em tempo real e em gerenciar o volume de dados de forma eficiente, o que pode envolver o desenvolvimento de algoritmos *out-of-core*. Tais algoritmos são projetados para situações onde a quantidade de dados a serem processados é maior do que a capacidade da memória principal [2]. Formas alternativas de armazenamento e compressão de dados estão aí envolvidas e podem ser seguidas como temas de pesquisa.

Uma alternativa recente e viável com o objetivo de diminuir o tempo de geração das imagens e manter a alta qualidade obtida, por exemplo, pelo uso do algoritmo de *ray casting*, é a utilização do mapeamento de textura por hardware para visualização volumétrica. Tanto a técnica de textura 3D quanto de multi-texturas podem ser usadas para extração de isosuperfícies e/ou para geração de imagens com diferentes níveis de transparência. Entretanto, a criação dessas texturas com base no volume de dados original é, por si só, uma tarefa complexa, pois envolve frequentemente, aspectos de armazenamento e compressão de dados (já mencionados acima) e, adicionalmente, a especificação de cores para intervalos de valores dos dados originais.

Nesse aspecto, nos últimos anos cresceu o desenvolvimento de técnicas automáticas ou semi-automáticas para a especificação de funções de transferência. Estas funções são

essenciais no processo de visualização, principalmente quando se trabalha com volumes de dados complexos, difíceis de visualizar e cujo conteúdo não é conhecido [30]. Por isso, a especificação através de tentativa e erro não é mais aceitável. Segundo Lorensen [30], as abordagens mais promissoras são as que pertencem à categoria centralizada nos dados, onde é feita uma análise das características do volume de dados para identificar o isovalor da(s) superfície(s) de interesse. De certa forma, isto significa a incorporação de funções de processamento e análise dos dados, como uma forma de segmentação semi-automática ou "inteligente", no processo de visualização.

Referências

[1] C.L. Bajaj, V. Pascucci, D.R. Schikore. The Contour Spectrum. In: IEEE Visualization 1997, *Proceedings...* p. 167–173, 539.

[2] K. Brodlie, J. Wood. Recent Advances in Volume Visualization. *Computer Graphics Forum*, vol. 20, no. 2, 2001. p. 125–148.

[3] E. Bullitt, S. Aylward. Volume Rendering of Segmented Tubular Objects. In: MICCAI 2001. *Proceedings...* p. 161–168.

[4] T.T. Elvins. A Survey of Algorithms for Volume Visualization. *ACM Computer Graphics*, vol. 26, no. 3, p. 194–201, Aug. 1992.

[5] K. Engel, M. Kraus, and T. Ertl. High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading. In Eurographics/SIGGRAPH Workshop on Graphics Hardware 2001 , Annual Conference Series, page 9. Addison-Wesley Publishing Company, Inc.

[6] S. Fang, T. Biddlecome, M. Tuceryan. Image-based transfer function design for data exploration in volume visualization. In: IEEE Visualization 1998. *Proceedings...* p. 319–326, 546.

[7] H. Fuchs, Z.M. Kedem, S.P. Uselton. Optimal Surface Reconstruction from Planar Contours. *Communications of the ACM*, vol. 20, no. 10, p. 693–702, 1977.

[8] R.C. Gonzalez, R.E. Woods. *Digital image processing*. Reading, MA: Addison-Wesley, 1993. 716 p.

[9] J. Hladùvka, A. König, E. Gröller. Curvature-Based Transfer Functions for Direct Volume Rendering. In: Bianca Falcidieno, editor, *Spring Conference on Computer Graphics 2000*, vol. 16, p. 58–65, May, 2000.

[10] A. Kanitsar, R. Wegenkittl, P. Felkel et al. Computed Tomography Angiography: A Case Study of Peripheral Vessel Investigation. In: IEEE Visualization 2001. October, 2001, San Diego, CA. *Proceedings...* p. 477–480.

[11] A.E. Kaufman, R. Bakalash, D. Cohen, R. Yagel. A survey of architectures for volume rendering. *IEEE Engineering in Medicine and Biology Magazine*, vol. 9, no. 4, Dec. 1990, p. 18–23.

[12] A.E. Kaufman. *Volume Visualization*. Los Alamitos, CA, IEEE Computer Society Press, 1991, 479 p.

[13] A.E. Kaufman, D. Cohen, R. Yagel. Volume Graphics. *IEEE Computer*, vol. 26, no. 7, July 1993. p. 51–64.

[14] E. Keppel. Approximating Complex Surfaces by Triangulation of Contour Lines. *IBM Journal of Research and Development*, vol. 19, no. 1, p. 2–11, Jan. 1975.

[15] G. Kindlmann and J.W. Durkin. Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering. In: IEEE Symposium on Volume Visualization, 1998. *Proceedings...* p. 79–86.

[16] G. Kindlmann. *Transfer Functions for Direct Volume Rendering*. In: IEEE Visualization 2001. Tutorial. Disponível em http://visual.nlm.nih.gov/tutorials/vis2001/ (Janeiro 2002).

[17] J. Kniss, G. Kindlmann, C. Hansen. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In: IEEE Visualization 2001. *Proceedings...* p. 255–262.

[18] A. König, E. Gröller. Mastering Transfer Function Specification by using VolumePro Technology. In: Tosiyasu L. Kunii, editor, *Spring Conference on Computer Graphics 2001*, vol. 17, p. 279–286, April, 2001.

[19] P. Lacroute, M. Levoy. Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation. In: SIGGRAPH - INTERNATIONAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, 1994. *Proceedings...* ACM Press. p. 451–458.

[20] M. Levoy. Volume Rendering - Display of Surfaces from Volume Data. *IEEE Computer Graphics & Applications*, vol. 8, no. 3, 1988, 29–37.

[21] M. Levoy. Efficient Ray-Tracing of Volume Data. *ACM Transactions on Graphics*, vol. 9, no. 3, 245–261.

[22] B. Lichtenbelt, R. Crane, S. Naqvi. *Introduction to Volume Rendering*. Upper Saddle River, NJ: Prentice Hall, 1998.

[23] B. Liu, G. Durdle, M. Margala, S. Juskiw. A simplified energy projection and applications of transfer function in isometric volume rendering. In: IEEE Canadian Conference on Electrical and Computer Engineering, 1999. *Proceedings...* p. 1728–1733 vol.3

[24] W.E. Lorensen, H.E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In: SIGGRAPH - INTERNATIONAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, 1987. *Proceedings...* ACM Press. p. 163–169.

[25] M. Meissner, U. Hoffmann and W. Strasser. Enabling Classification and Shading for 3D Texture Mapping based Volume Rendering using OpenGL and Extensions. In: IEEE Visualization 1999, San Francisco, USA. *Proceedings...* p. 207–14

[26] M. Meissner, H. Pfister, R. Westermann, C. Wittenbrink. *Volume Visualization and Volume Rendering Techniques*. In: EUROGRAPHICS Conference 2000. Tutorial. Disponível em http://www.gris.uni-tuebingen.de/ meissner/tutorials/tutorial.pdf (Julho de 2002).

[27] K. Mueller, T. Möller, R. Crawfis. Splatting Without the Blur. In: IEEE Visualization 1999, San Francisco, CA. *Proceedings...* p. 363–370.

[28] R. Mullick, N. Bryan, J. Butman. Confocal Volume Rendering: Fast Segmentation-Free Visualization of Internal Structures. In: Medical Imaging 2000 - Image Processing, SPIE, 2000, San Diego, California. *Proceedings....*

[29] R. Osborne, H. Pfister, H. Lauer et al. EM-Cube: An Architecture for Low-Cost Real-Time Volume Rendering. In: SIGGraph/Eurographics Workshop on Graphics Hardware 1997, Los Angeles, CA. *Proceedings...*

[30] H. Pfister, B. Lorensen, C. Bajaj et al. The Transfer Function Bake-Off. *IEEE Computer Graphics & Applications*, vol. 21, no. 3, May/June 2001. p. 16–22.

[31] H. Pfister, J. Hardenbergh, J. Knittel et al. The VolumePro Real-Time Ray-Casting System. In: SIGGRAPH - INTERNATIONAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, 1999. *Proceedings...* ACM Press. p. 251–260.

[32] H. Ray, H. Pfister, D. Silver, T.A. Cook. Ray casting architectures for volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 3, July-Sept. 1999. p. 210–223.

[33] C. Rezk-Salama, K. Engel et al. Interactive Volume Rendering on Standard PC Graphics Hardware Using Multi-Textures and Multi-Stage Rasterization. In: SIGGraph/Eurographics Workshop on Graphics Hardware 2000. *Proceedings...*

[34] R.A. Robb. *Three-Dimensional Biomedical Imaging: principles and practice*. New York, NY, John Wiley & Sons, 1998.

[35] M.R.M. Silva. *Alternativas de Visualização de Volumes Baseadas em Ray Casting*. Porto Alegre: Instituto de Informática/UFRGS, 2000, Dissertação de Mestrado.

[36] J.K. Udupa. D. Odhner. Shell Rendering. *IEEE Computer Graphics & Applications*, vol. 13, no. 6, 1993, 58–67.

[37] C. Upson; M. Keeler. V-Buffer: Visible Volume Rendering. In: SIGGRAPH - INTERNATIONAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, 1988. *Proceedings...* ACM Press. p. 59–64.

[38] D. Weinstein. Scanline Surfacing: Building Separating Surfaces from Planar Contours. IEEE Visualization 2000, *Proceedings...* p. 283–289.

[39] L. Westover. Interactive Volume Rendering. In: Workshop on Volume Visualization, 1989, Chapel Hill, N.C. *Proceedings...* University of North Carolina Press. p. 9–16.

[40] L. Westover. Footprint Evaluation for Volume Rendering. In: SIGGRAPH - INTERNATIONAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, 1990. *Proceedings...* ACM Press. p. 367–376.

[41] R. Westermann, T. Ertl. Efficiently using Graphics Hardware in Volume Rendering Applications. In: SIGGRAPH - INTERNATIONAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, 1998. *Proceedings...* ACM Press. p. 169–177.

[42] J. Wilhelms, A. Van Gelder. A Coherent Projection Approach for Direct Volume Rendering. In: SIGGRAPH - INTERNATIONAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, 1991. *Proceedings...* ACM Press. p. 275–284.

[43] S.Y. Yen, S. Napel, G.D. Rubin. Fast Sliding Thin Slab Volume Visualization. In: IEEE Symposium on Volume Visualization, 1996. *Proceedings...* p. 79–86.

[44] K.J. Zuiderveld. *Visualization of Multimodality Medical Volume Data using Object-Oriented Methods*. Netherlands: Universiteit Utrecht, 1995. Ph.D. Thesis.