UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Especificação de Funções de Transferência para Visualização Volumétrica

JOÃO LUIS PRAUCHNER

Dissertação apresentada como requisito
parcial para obtenção do grau de
Mestre em Ciência da Computação

Profa. Dra. Carla Maria Dal Sasso Freitas
Orientadora
Prof. Dr. João Luiz Dihl Comba
Co-orientador

Porto Alegre, julho de 2005

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Prauchner, João Luis

Especificação de Funções de Transferência para Visualização Volumétrica / João Luis Prauchner. – Porto Alegre: Programa de Pós-Graduação em Computação, 2005.

67f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2005. Orientadora: Carla M. D. S. Freitas; Co-orientador: João L. D Comba.

1. Visualização volumétrica. 2. Função de transferência. I. Freitas, Carla Maria Dal Sasso. II. Comba, João Luiz Dihl. III. Título.

AGRADECIMENTOS

O mestrado representou um período em que muitas coisas boas e contrutivas aconteceram em minha vida, durante o qual muitas pessoas participaram e ajudaram.

Agradeço aos meus pais João Renato e Luci Ida e meus irmãos Luciana e Renato pelo amor, carinho, apoio e incentivo que recebi em todos os momentos de minha vida.

Agradeço a minha orientadora Carla Dal Sasso e co-orientador João Comba pelo incentivo, apoio e ensinamentos transmitidos durante o desenvolvimento dos trabalhos.

Enfim gostaria de agradecer a todos os colegas, em especial a Carlos Dietrich, Christian Pagot, Diego Martins, Luciano Silva, Rodrigo Luque, Leandro Fernandes, Eduardo Pons e Carlos Scheidegger, também aos funcionários e professores do Instituto de Informática, por toda a ajuda, idéias, sugestões e lições que me deixaram.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

2D	bi-dimensional.
3D	tri-dimensional.
FT	Função de Transferência.
GPU	*Graphics Processor Unit* (Unidade de Processamento Gráfico).
GLUI	*Grapchis Library User Interface.*
HLS	*Hue, Lightness, Saturation* (modelo de cores matiz, iluminação e saturação).
HLSL	*High Level Shading Language* (Linguagem de Tonalização de Alto Nível).
HSV	*Hue, Saturation, Value* (modelo de cores matiz, saturação e valor).
OpenGL	*Open Graphics Library*.
PC	Computador Pessoal (*Personal Computer*).
RGB	*Red*, *Green*, *Blue* (canais de cor vermelho, verde e azul).
RM	Ressonância Magnética (*Magnetic Resonance*).
TC	Tomografia Computadorizada (*Computed Tomography*).
VGA	*Video Graphics Array*.

LISTA DE SÍMBOLOS

$\alpha[i]$	opacidade (grandeza escalar) amostrada na posição i do espaço.
v	valor escalar de um voxel em um volume de dados discreto.
$f'(x)$	primeira derivada de uma função $f(x)$.
$f''(x)$	segunda derivada de uma função $f(x)$.
$g(v)$	média dos valores de f' amostrados para um voxel v em um histograma 3D.
$h(v)$	média dos valores de f'' amostrados para um voxel v em um histograma 3D.

LISTA DE FIGURAS

RESUMO

Técnicas de visualização volumétrica direta são utilizadas para visualizar e explorar volumes de dados complexos. Dados volumétricos provêm de diversas fontes, tais como dispositivos de diagnóstico médico, radares de sensoriamento remoto ou ainda simulações científicas assistidas por computador. Um problema fundamental na visualização volumétrica é a especificação de Funções de Transferência (FTs) que atribuem cor e opacidade aos valores escalares que compõem o volume de dados. Essas funções são importantes para a exibição de características e objetos de interesse do volume, porém sua definição não é trivial ou intuitiva. Abordagens tradicionais permitem a edição manual de pontos de controle que representam a FT a ser utilizada no volume. No entanto, essas técnicas acabam conduzindo o usuário a um processo de "tentativa e erro" para serem obtidos os resultados desejados. Considera-se também que técnicas automáticas que excluem o usuário do processo não são consideradas as mais adequadas, visto que o mesmo deve possuir algum controle sobre o processo de visualização. Este trabalho apresenta uma ferramenta semi-automática e interativa destinada a auxiliar o usuário na geração de FTs de cor e opacidade. A ferramenta proposta possui dois níveis de interação com o usuário. No primeiro nível são apresentados várias FTs candidatas renderizadas como *thumbnails* 3D, seguindo o método conhecido como *Design Galleries* (MARKS et al., 1997). São aplicadas técnicas para reduzir o escopo das funções candidatas para um conjunto mais razoável, sendo possível ainda um refinamento das mesmas. No segundo nível é possível definir cores para a FT de opacidade escolhida, e ainda refinar essa função de modo a melhorá-la de acordo com as necessidades do usuário. Dessa forma, um dos objetivos desse trabalho é permitir ao usuário lidar com diferentes aspectos da especificação de FTs, que normalmente são dependentes da aplicação em questão e do volume de dados sendo visualizado. Para o *rendering* do volume, são exploradas as capacidades de mapeamento de textura e os recursos do hardware gráfico programável provenientes das plácas gráficas atuais visando a interação em tempo real. Os resultados obtidos utilizam volumes de dados médicos e sintéticos, além de volumes conhecidos, para a análise da ferramenta proposta. No entanto, é dada ênfase na especificação de FTs de propósito geral, sem a necessidade do usuário prover um mapeamento direto representando a função desejada.

Palavras-chave: visualização volumétrica, função de transferência, hardware gráfico, visualização interativa.

TRANSFER FUNCTION SPECIFICATION FOR VOLUMETRIC VISUALIZATION

ABSTRACT

Direct volume rendering techniques are used to visualize and explore large scalar volumes. Volume data can be acquired from many sources including medical diagnoses scanners, remote sensing radars or even computer-aided scientific simulations. A key issue in volume rendering is the specification of Transfer Functions (TFs) which assign color and opacity to the scalar values which comprise the volume. These functions are important to the exhibition of features and objects of interest from the volume, but their specification is not trivial or intuitive. Traditional approaches allow the manual editing of a graphic plot with control points representing the TF being applied to the volume. However, these techniques lead the user to an unintuitive trial and error task, which is time-consuming. It is also considered that automatic methods that exclude the user from the process should be avoided, since the user must have some control of the visualization process. This work presents a semi-automatic and interactive tool to assist the user in the specification of color and opacity TFs. The proposed tool has two levels of user interaction. The first level presents to the user several candidate TFs rendered as 3D thumbnails, following the method known as Design Galleries (MARKS et al., 1997). Techniques are applied to reduce the scope of the candidate functions to a more reasonable one. It is also possible to further refine these functions at this level. In the second level is permitted to define and edit colors in the chosen TF, and refine this function if desired. One of the objectives of this work is to allow users to deal with different aspects of TF specification, which is generally dependent of the application or the dataset being visualized. To render the volume, the programmability of the current generation of graphics hardware is explored, as well as the features of texture mapping in order to achieve real time interaction. The tool is applied to medical and synthetic datasets, but the main objective is to propose a general-purpose tool to specify TFs without the need for an explicit mapping from the user.

Keywords: volume rendering, transfer function, graphics hardware, interactive visualization

1 INTRODUÇÃO

1.1 Visualização Volumétrica

O desenvolvimento de aplicações de visualização visa fornecer aos usuários ferramentas que auxiliem nas tarefas de análise, exibição e exploração de grandes conjuntos de dados (MANSSOUR, 2002). Em particular, a visualização volumétrica tem como objetivo permitir a análise visual e a exploração de dados volumétricos (3D). Suas técnicas fornecem mecanismos para exibir e explorar o interior dos dados volumétricos, possibilitando a identificação de regiões e estruturas internas (KAUFMAN et al., 1993). As informações normalmente são armazenadas em uma grade regular, mas grades retilíneas, curvilíneas e irregulares também podem ser utilizadas. Esses dados podem ser obtidos através de diversas fontes, tais como simulações científicas, equipamentos de aquisição de imagens médicas (como tomografia por raio X, ressonância magnética ou ultrassom) ou, ainda radares. Esses volumes podem ainda ser multidimensionais, quando variam com o tempo, ou quando vários valores são amostrados por unidade espacial.

Entre as várias áreas do conhecimento que se beneficiam das aplicações de visualização volumétrica podemos 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 planejar uma cirurgia ou simulá-la num treinamento médico. 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, tais aplicações aparecem na modelagem e animação de estruturas moleculares, facilitando a visualização da interação entre as moléculas. Com o advento do hardware gráfico programável e o crescente aumento no desempenho das placas gráficas atuais, as técnicas de visualização volumétrica também estão sendo utilizadas para novas aplicações na Computação Gráfica, como a geração de efeitos volumétricos para cenas sintéticas em tempo real (HARRIS et al., 2001), como ilustrado na figura 1.1.

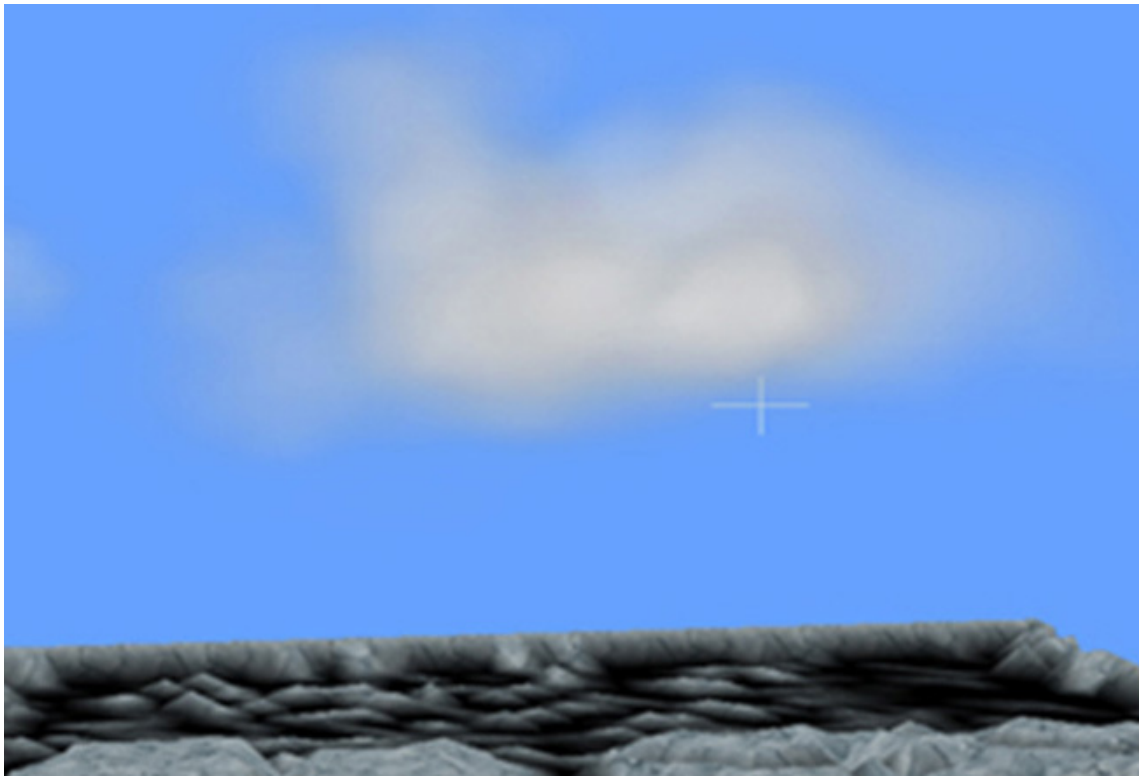


Figura 1.1: Cena sintética renderizada em tempo real. A nuvem presente na cena é um volume de dados exibido através de visualização volumétrica direta (PRAUCHNER et al., 2004).

Na literatura encontra-se, freqüentemente, as técnicas de visualização volumétrica classificadas como técnicas de extração de isosuperfícies ou de visualização direta (KAUFMAN, 1991). Na classe de técnicas de extração de isosuperfícies, a visualização é feita através da construção de uma malha de polígonos que representa uma superfície de contorno extraída do volume de dados. Esses polígonos são então exibidos utilizando o *pipeline* gráfico tradicional. Entre os algoritmos de visualização através de superfícies destacam-se: conexão de contornos (KEPPEL, 1975; FUCHS et al., 1977) e cubos marchantes (LORENSEN et al., 1987). A outra classe principal é denominada visualização direta de volume, onde o *rendering* das amostras do volume de dados (denominados voxels) é realizado sem a extração de geometrias intermediárias. As técnicas de visualização volumétrica direta ainda podem ser classificadas, segundo (KAUFMAN, 1991; BRODLIE et al., 2001), em: *object-order* ou *forward projection* (GELDER et al., 1996; ENGEL et al., 2002), que envolve o mapeamento de amostras de dados no plano da imagem; *image-order* ou *backward projection* (WITTENBRINK et al., 1998; KRUEGER et al. 2003), que determina para cada pixel do plano da imagem quais são as amostras que contribuem para sua intensidade; e *domain-based*, quando os dados 3D são transformados para outro domínio, como frequência ou *wavelet* (WESTENBERG et al., 2000).

Um dos principais problemas enfrentados na visualização volumétrica é a conversão entre os valores de dados escalares originais e atributos visuais, como cor e opacidade. Esse processo é geralmente feito através do uso de funções de transferência ou FTs. Essas funções são particularmente importantes para o resultado e a qualidade das imagens exibidas, pois representam uma classificação dos voxels que compõem o volume. No entanto, 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, sendo que um de seus problemas é a forte dependência da aplicação em questão.

No PPGC, visualização volumétrica direta já foi tema de dissertações de mestrado (SILVA, 2000; SILVA, 2003; DIETRICH, 2004) e tese de doutorado (MANSSOUR, 2002). Nesses trabalhos, a questão das funções de transferência não foi tratada com a profundidade necessária. Silva (SILVA, 2000) implementou visualização por *ray casting*, com funções de transferência na forma de gráficos inicializados com edição por parte do usuário; Manssour (MANSSOUR, 2002) também implementou *ray casting* com o objetivo de visualização de dados multimodais, mas com funções de transferência também especificadas de forma manual, através da edição indireta da função. Silva (SILVA, 2003) desenvolveu métricas para a avaliação da qualidade de imagens geradas por *ray casting* quando aplicado a volumes de dados médicos. Já Dietrich (DIETRICH, 2004) implementou ferramentas para a visualização de volumes de dados médicos, utilizando uma função de transferência configurada pelo radiologista nos dispositivos de aquisição desses volumes.

Este trabalho contribui para esta área com a proposta e desenvolvimento de uma ferramenta para especificação de funções de transferência que maximiza o uso automático da informação que pode ser extraída do volume, mas permite ao usuário o refinamento da função dependendo dos resultados automáticos encontrados.

1.2 Objetivos e Contribuição

No contexto exposto acima, a dissertação tem como objetivo propor uma ferramenta interativa para a especificação de funções de transferência para visualização volumétrica direta utilizando recursos de programação das placas gráficas. Essa ferramenta permite a especificação e o refinamento de FTs em dois níveis de interação, além da visualização em tempo real de imagens volumétricas de diferentes ramos de aplicação.

Considerando que a tarefa de especificar funções de transferência que gerem imagens de qualidade tem sido listada entre os dez maiores problemas da visualização volumétrica (BOTHA et al., 2002), as principais contribuições deste trabalho em relação a publicações anteriores são:

a) A geração automática de funções de transferência iniciais com base em informações extraídas do volume;
b) O algoritmo denominado dispersão baseada nas bordas (*edge-based dispersion*) para a dispersão e variabilidade das FTs geradas automaticamente, descrito na seção 4.4.1;
c) A implementação de uma técnica de navegação interativa no espaço das TFs iniciais exibidas como *thumbnails* 3D, descrita na seção 4.4.2;
d) A geração e refinamento das funções iniciais com resultados exibidos em tempo real, descrita na seção 4.4.3.

1.3 Organização do Texto

A visualização volumétrica é um tópico bastante amplo da Computação Gráfica. No capitulo 2 são revistos conceitos sobre as técnicas existentes na literatura de visualização volumétrica em geral e também sobre a definição de FTs. Também é abordada a utilização dos recursos do hardware gráfico atual em aplicações de visualização.

No Capítulo 3 são discutidos os principais trabalhos existentes na literatura sobre especificação de FTs relacionados a este estudo. São discutidas as vantagens e desvantagens de cada método.

No Capítulo 4 é proposta a técnica de especificação de FTs em dois níveis de interação. Cada nível é detalhado com relação a decisões de implementação e funcionalidades existentes. Também é detalhada a implementação da ferramenta.

No Capítulo 5 são mostrados os resultados da aplicação da ferramenta proposta sobre volumes de dados médicos e sintéticos. Primeiro é vista a importância da ferramenta na visualização e exploração das imagens 3D. Na seqüência, podem ser observados os resultados da aplicação da ferramenta proposta neste estudo em comparação com outras interfaces existentes.

No Capítulo 6, são discutidas as principais questões envolvidas no projeto e implementação da ferramenta proposta. Também são sugeridos trabalhos futuros a serem realizados com o objetivo de melhorar a eficiência da ferramenta desenvolvida.

2 VISUALIZAÇÃO DIRETA DE VOLUME

2.1 Processo básico de visualização direta

A visualização volumétrica direta dispensa o uso de primitivas geométricas para a geração da imagem pois consiste em tomar os voxels do volume e projetá-los diretamente em pixels na imagem. Neste caso, numa etapa de classificação, é utilizada uma função de transferência, que corresponde ao mapeamento dos valores dos voxels (densidade do tecido em uma tomografia, por exemplo) para propriedades visuais, tais como cor e opacidade. A visualização das estruturas de interesse presentes 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.

As técnicas de visualização direta de volume possuem um alto custo computacional, pois normalmente envolvem 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* (LEVOY, 1988, LEVOY 1990, PFISTER et al., 1999, SILVA 2000, SILVA 2003), *splatting* (WESTOVER 1989, MUELLER et al., 1990; WESTOVER 1990, MUELLER et al., 1990; HOPF et al., 2003), *shear-warp* (LACROUTE et al., 1994, HAUSER et al., 2001), *shell rendering* (UDUPA et al., 1993), *cell-projection* (WILHELMS et al., 1991) e *V-Buffer* (UPSON et al., 1988). A figura 2.1 mostra, de um modo geral, as etapas do processamento realizado pela visualização direta de volume.

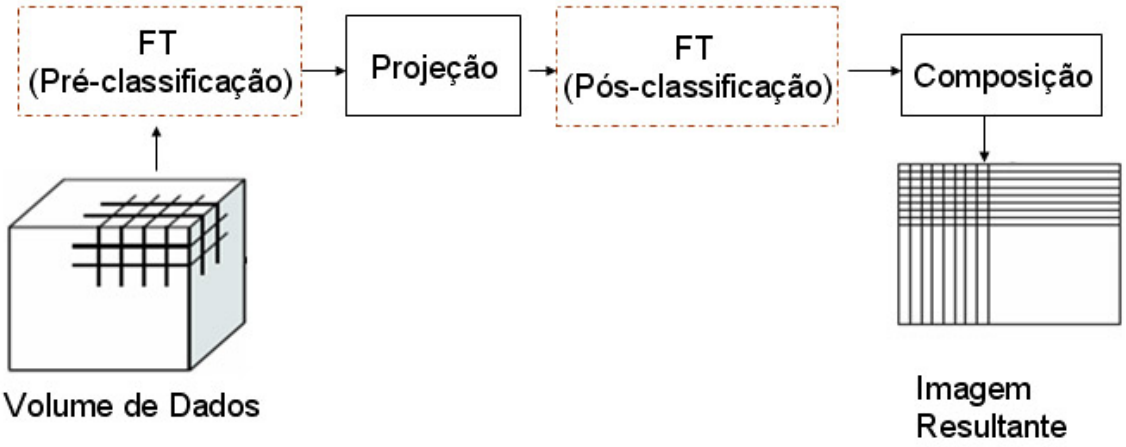


Figura 2.1: *Pipeline* básico da visualização direta de volume.

Na figura 2.1, a etapa inicial corresponde à obtenção dos dados que formam o volume. A seguir, é feita a classificação dos dados (tarefa executada pelas funções de transferência), juntamente com cálculos adicionais, como modelos de iluminação e tonalização (MAX, 1995). Então os dados passam por uma interpolação, feita com o objetivo de compensar a discretização sofrida pelos mesmos (que normalmente são extraídos de um espaço contínuo) em sua aquisição. Finalmente os dados passam por transformações de projeção e são compostos no *frame buffer*, sendo então exibidos na tela como pixels.

Alguns autores 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. 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, a abordagem é conhecida como "pré-classificação" ou *pre-shaded volume rendering* (WITTENBRINK et al., 1998; ENGEL et al., 2001). O processo no qual os valores originais dos voxels são interpolados para depois serem executadas a classificação e a iluminação é denominado "pós-classificação" ou *post-shaded volume rendering* (MEISSNER et al., 1999). O diagrama da figura 2.1 apresenta ambas as formas de visualização e a ordem de cada etapa.

Apesar de serem fundamentais para a visualização direta de volumes, as funções de transferência não são facilmente especificadas, pois apresentam uma forte dependência à natureza dos dados que compõem o volume. Além disso, essas funções idealmente precisam ser capazes de realçar informações de interesse nos dados, além de descartar dados irrelevantes. Na literatura vêm sendo pesquisadas técnicas para geração dessas funções, a fim de tornar a tarefa mais prática e simples. Na seção a seguir é feita uma descrição conceitual do que são funções de transferência (FTs) com base no trabalho de Manssour (MANSSOUR, 2002) e no capítulo 3 é mostrado um estudo sobre as principais técnicas existentes para a especificação das mesmas.

2.2 Funções de Transferência

A função de transferência é responsável pela atribuição de propriedades visuais aos valores originais do volume de dados que será visualizado (PFISTER et al., 2001). As propriedades mais utilizadas são opacidade e cor, mas emissão e índice de refração também podem ser utilizadas (KINDLMANN et al., 2002). As FTs são usadas no estágio denominado classificação, nos algoritmos de visualização volumétrica, e permitem explorar as estruturas existentes nos volumes de dados sem a necessidade de definir a forma ou extensão das regiões de interesse.

Em uma FT típica, um valor de opacidade é geralmente associado a cada voxel do volume, a fim de descrever a quantidade de energia luminosa absorvida por esse elemento. Basicamente, esse valor indica se é possível ou não enxergar através de um voxel, permitindo assim a definição de estruturas transparentes, semitransparentes e opacas. A função para atribuir cor é necessária porque a maioria dos volumes de dados não possui valores de cor associados a cada voxel (LICHTENBELT et al., 1998).

Especificar funções de transferência que gerem imagens de qualidade figura entre os dez maiores problemas da visualização volumétrica (BOTHA et al., 2002). Um problema, por exemplo, é que normalmente os valores de opacidade são definidos através da edição de gráficos que possuem pontos de controle alteráveis. Esses gráficos representam uma FT, no

entanto, especificá-las desta maneira tende a ser uma tarefa demorada e realizada através de "tentativa e erro", pois pequenas mudanças na função podem causar drásticas mudanças na imagem gerada (KINDLMANN et al., 2002). Além disso, se a aplicação em questão não apresentar velocidade interativa (o que é comum em aplicações de visualização volumétrica), a tarefa fica ainda mais demorada.

Devido à sua complexidade, nos últimos anos houve um aumento na pesquisa de técnicas automáticas e semi-automáticas para a criação destas funções, e no desenvolvimento de interfaces interativas mais intuitivas para auxiliar nesta tarefa. Isto pode ser facilmente observado pelos vários trabalhos recentes encontrados na literatura (PFISTER et al., 1996; BAJAJ et al., 1997; MARKS et al., 1997; KINDLMANN et al. 1998; HLADUVKA et al., 2000; KONIG et al., 2001; PFISTER et al. 2001; BOTHA et al. 2002; KNISS et al., 2002; SRIVASTAVA et al., 2002; HONIGMANN et al., 2003; KINDLMANN et al., 2003; KNISS et al., 2003; TZENG et al., 2003).

Segundo Lorensen (PFISTER et al. 2001), técnicas automáticas ou que requerem muita interação com o usuário não são as mais adequadas. Quando há necessidade de grande interação, como por exemplo nos gráficos definidos através de "tentativa e erro", a tarefa de encontrar uma função de transferência adequada pode levar muito tempo. Por outro lado, o usuário não deve ser afastado do processo de especificação destas funções, uma vez que o objetivo das técnicas de visualização é permitir a exploração e o entendimento do volume de dados, e não gerar imagens bonitas. Além disso, técnicas automáticas tendem a perder detalhes que apenas um observador humano poderia notar.

2.2.1 Funções de Transferência de Opacidade

As funções de transferência mais simples especificam a opacidade apenas de acordo com a intensidade do voxel, seguindo a fórmula:

$$\alpha = f(v) \qquad \text{Equação 2.1}$$

onde v é o valor escalar ou intensidade de um voxel e α é a opacidade atribuída a esse voxel através da FT f. Atualmente existe um conjunto padrão de funções de transferência de opacidade que normalmente é utilizado nos sistemas de visualização volumétrica direta. Entre estas funções destacam-se: rampa, trapézio, bloco, exponencial e linear.

Numa função linear, quanto maior for a intensidade do voxel, maior é o valor de opacidade que ele recebe. 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. A função chamada de "rampa" por König e Gröller (KONIG et al., 2001), é uma variação da função linear, que ocorre quando o valor de intensidade do voxel que possui opacidade 1 (máxima) não é o maior, mas sim o central. Outro exemplo de função de transferência utilizada devido à sua simplicidade é a função exponencial. Já a FT do tipo bloco normalmente é utilizada para atribuir um mesmo valor de opacidade a um conjunto de voxels pertencentes a um determinado intervalo. A Figura 2.2 mostra exemplos dos gráficos das funções mencionadas.

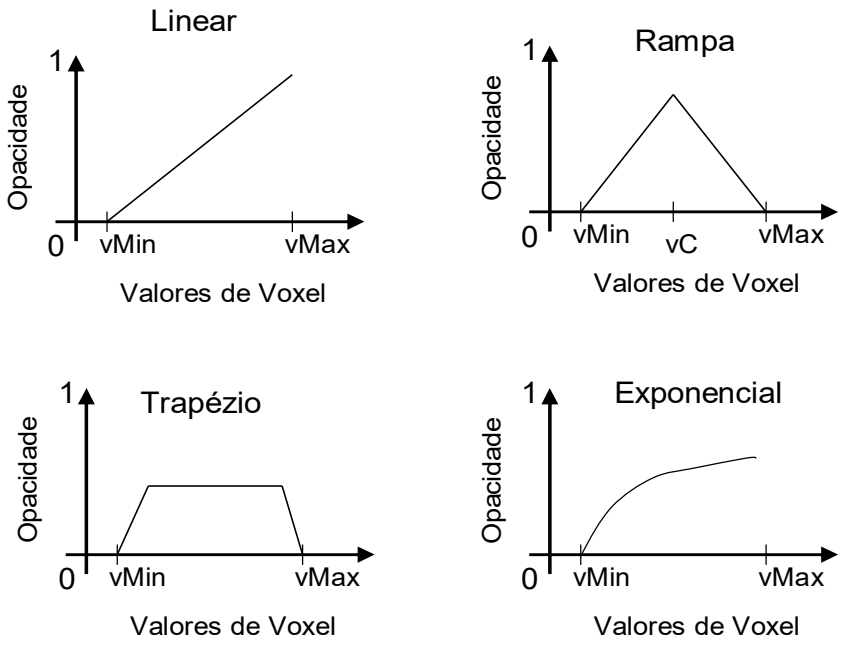


Figura 2.2: Gráficos de FTs de opacidade dos tipos linear, rampa, trapézio e exponencial.

Nos gráficos, *vMin* é o valor mínimo que um voxel pode ter em um volme de dados arbitrário. Analogamente, *vMax* é o valor máximo e *vC* é o valor central. Normalmente, uma FT pode ser armazenada em uma *lookup table* ou tabela de classificação, que pode ser interpretada como um vetor cujo índice indica o valor de intensidade e o conteúdo o valor de opacidade.

Além da utilização das funções descritas anteriormente, outra forma usual de especificar uma função de transferência é através da edição manual de um gráfico com pontos de controle que representam o mapeamento de um voxel para um valor de opacidade, como mencionado na seção 2.2. Para a construção da *lookup table* que representam as FTs definidas dessa forma, são inseridos os pontos de controle na tabela e as demais posições, entre os pontos, são preenchidas através da utilização de uma função de interpolação, tal como a função linear. A Figura 2.3 ilustra esse tipo de abordagem.

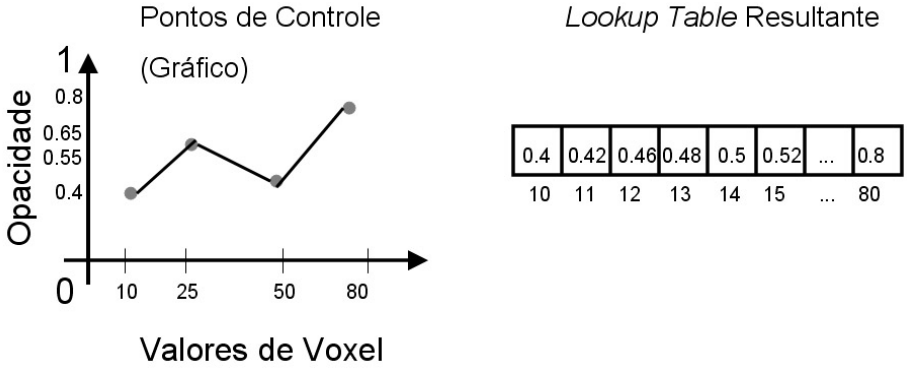


Figura 2.3: Gráfico de uma FT definida através de 4 pontos de controle (à esquerda). Os pontos são então interpolados para gerar a tabela de classificação resultante (à direita).

2.2.2 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 volumétrica direta, outros modelos de cor, tais como *Hue-Lightness-Saturation* (HLS) e *Hue-Saturation-Value* (HSV), também são utilizados.

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 utilidade, é preciso ter cuidado na definição das tabelas de cores, pois da mesma maneira que o seu uso pode melhorar a visualização, também pode prejudicar, visto que cor é considerado um atributo subjetivo em uma imagem.

2.3 Visualização direta de volumes utilizando hardware gráfico

2.3.1 Hardware Gráfico

Na ultima década houve um avanço surpreendente na capacidade do hardware gráfico (FERNANDO et al., 2003). Como resultado dos esforços de aumentar o poder de processamento das placas gráficas, surgiram gerações de GPUs (Graphics Processor Units) que englobam praticamente todo o *pipeline* gráfico em um único *microchip.* As GPUs possuem um poder de processamento muito superior ao da CPU, no entanto não são de propósito geral como a CPU. Antes do surgimento das GPUs, o hardware gráfico era restrito a estações computacionais de custo elevado, desenvolvidas principalmente por empresas como a SGI® e Evans&Sutherland®. Esses sistemas foram importantes pois introduziram muitos conceitos adotados nas GPUs de hoje, como transformações de vértice e mapeamento de texturas. No entanto, devido ao custo, não tiveram a popularização obtida pelas GPUs. O avanço tecnológico das GPUs foi movido basicamente pela aumento na complexidade das

técnicas de computação gráfica, e também pelas indústrias cinematográficas e de entretenimento, que desejavam produzir imagens sintéticas de melhor qualidade.

A primeira geração de GPUs (por volta de 1998) implementava algumas funções do *pipeline* gráfico, como a rasterização de polígonos e a aplicação de uma ou mais texturas. Exemplos de placas gráficas dessa geração incluem TNT2, da nVIDIA®, Rage, da ATI® e Voodoo3, da 3dfx®. Apesar de retirarem da CPU o trabalho de atualizar os pixels do *frame buffer*, essas placas não apresentavam funções de transformação de vértices, então executadas pela CPU. Além disso, possuíam um conjunto limitado de operações para combinar texturas e calcular a cor final do pixel.

A segunda geração de GPUs (1999-2000) passou a incluir a transformação e iluminação rápida de vértices, retirando esse trabalho da CPU. Exemplos de placas gráficas dessa geração incluem as GeForce 256 e GeForce2, da nVIDIA®, Radeon 7500, da ATI® e Savage3D, da S3®. Além de implementarem praticamente todo o *pipeline* gráfico tradicional, esses dispositivos eram mais configuráveis (várias características do *pipeline* fixo podem ser configuradas pela aplicação). No entanto, ainda havia muitas limitações a serem exploradas. Como resultado disso, em 2001, surgiu a terceira geração de GPUs, que introduziu os recursos de programabilidade no nível de vértice. Exemplos dessas placas incluem GeForce3 e GeForce4, da nVIDIA®, Xbox, da Microsoft® e Radeon 8500, da ATI®. Dessa forma, o *pipeline* antes fixo passou a ter a capacidade de executar algoritmos (chamados de *shaders*) específicos para o processamento de vértices, além de possuir as funcionalidades já existentes em OpenGL. OpenGL é uma interfaçe de programação desenvolvida em 1992 pela SGI® que veio a se tornar o primeiro padrão mundialmente reconhecido para a programação de aplicações gráficas. Seu objetivo é prover um conjunto de recursos, sequencialmente utilizados , para a visualização de imagens e primitivas geométricas, como ilustrado na Figura 2.4.

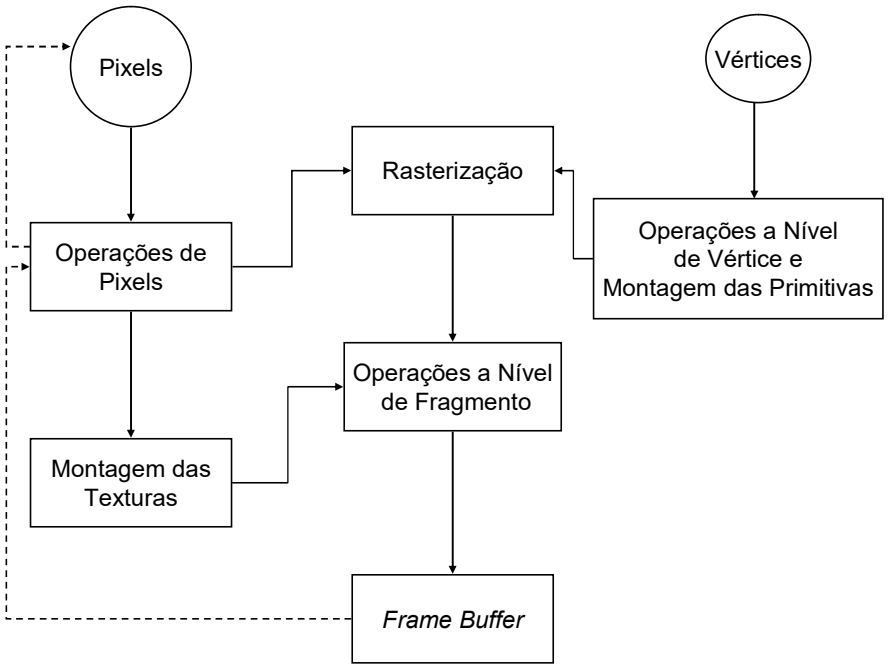


Figura 2.4: *Pipeline* de visualização da OpenGL (baseado em SHREINER, 1999).

Outra interface de programação bastante conhecida é o DirectX, desenvolvido pela Microsoft. Apesar de ser diferente da OpenGL, ambos especificam um *pipeline* para sistemas gráficos de modo geral. No diagrama da Figura 2.4, as informações principais são relativas aos vértices e aos pixels (texturas), sendo que os vértices geralmente são descritos em um espaço 3D, e as texturas como imagens 2D a serem mapeadas nas faces dos polígonos. A tarefa principal do *pipeline* é, a partir de um ponto de vista específico, projetar o universo 3D em uma imagem bidimensional, armazenada no *frame buffer*. Os vértices e as texturas passam, então, por etapas de transformação (que basicamente mudam o sistema de coordenadas do universo para o sistema de coordenadas do observador ou câmera). Após a montagem das primitivas, a geometria passa pelo estágio de rasterização, que divide os polígonos em fragmentos (candidatos a pixels no frame buffer). As texturas são então aplicadas e interpoladas para cada fragmento considerando suas coordenadas de textura, e a imagem final é composta no *frame buffer.* Os pixels do *frame buffer* ainda podem ser lidos e processados pela aplicação quando há necessidade para tal. No entanto essa operação é relativamente custosa.

A quarta geração de GPUs (de 2002 ao início de 2004) aumentou consideravelmente os resursos de programação em hardware. Exemplos dessas placas incluem GeForce FX da nVIDIA® e as Radeon 9700 e Radeon 9800 da ATI®. Nessa geração foi introduzido o controle programável no nível de vértice e fragmento. Dessa forma, as operações complexas de vértices e de coloração de fragmentos foram transportadas da CPU para a GPU, provendo mais liberdade para o desenvolvimento de novos efeitos de *rendering* em tempo real. Além disso, com os novos recursos programáveis, a GPU passou a ser explorada em diversas aplicações de propósito geral (além da geração de imagens sintéticas) como sendo um co-processador aritmético da CPU, auxiliando-a no cálculo de operações matemáticas complexas.

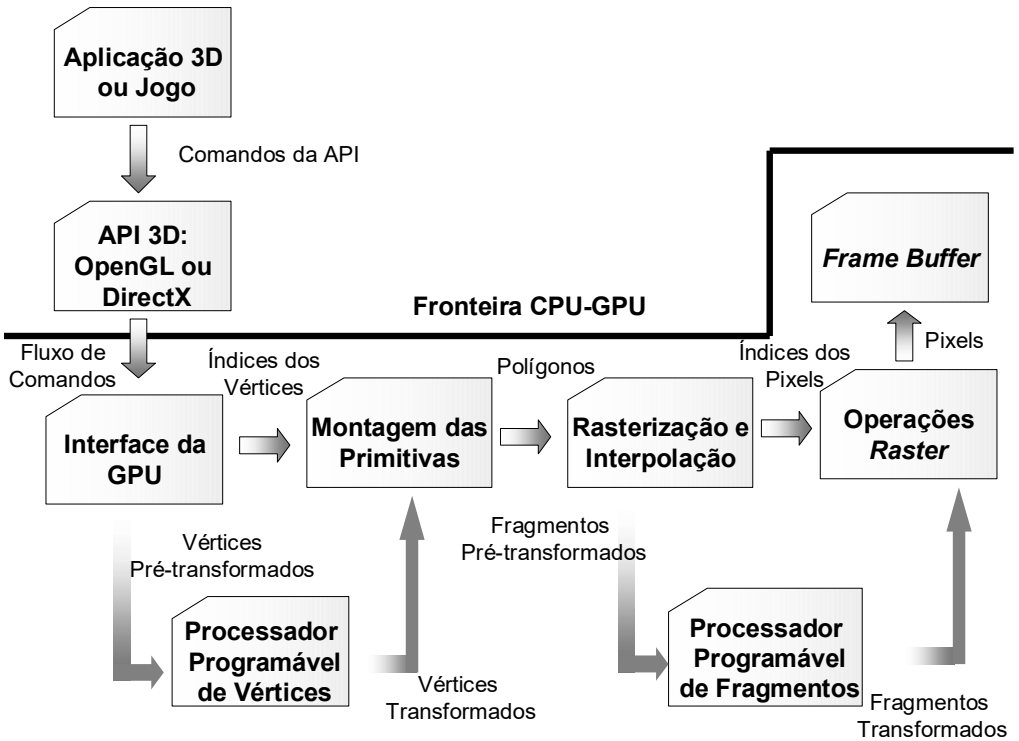


Figura 2.5: Diagrama de blocos do *pipeline* gráfico introduzido na quarta geração de GPUs (baseado em FERNANDO et al., 2003).

No entanto, a aplicação em questão necessita da adaptação ao modelo de *pipeline* vetorial presente na GPU a fim de se obterem resultados que justifiquem seu uso. A Figura 2.5 mostra o diagrama de blocos do *pipeline* gráfico programável introduzido na quarta geração de GPUs. Juntamente com o novo hardware, diversas linguagens (*C for Graphics* da nVIDIA®, *High Level Shading Language* da Microsoft®, OpenGL 2.0 *Shading Language*, etc) foram desenvolvidas a fim de oferecem formas de traduzir algoritmos em seqüências de instruções que a GPU possa executar.

No diagrama da Figura 2.5, os processadores programáveis executam os *shaders* de vértices e fragmentos. A CPU, através da aplicação gráfica em questão, transmite à GPU um fluxo de comandos de *rendering* e informações sobre vértices e texturas (como posição do vértice, cor, coordenadas de textura, normais). Os vértices passam por transformações geométricas, que podem ser fixas ou programáveis através de *shaders.* Após o processamento dos vértices, as primitivas geométricas ou polígonos são construídos e enviados ao próximo estágio, a rasterização. Esse estágio determina quais fragmentos cada polígono gera na cena sendo visualizada. As coordenadas de textura (e cores) do polígono são interpoladas a fim de que cada fragmento receba um valor de cor proveniente da unidade de textura sendo usada no momento (ou um valor de cor especificado pela aplicação). Após esse estágio, os fragmentos passam pelo processador de fragmentos, que pode executar um *shader* capaz de modificar a sua cor final, ou seguir adiante, sem sofrer modificações. O último estágio consiste nas operações *raster*. Essas operações basicamente executam testes para verificar se o fragmento deve ser escrito no *frame buffer* ou simplesmente descartado. Exemplos dessas operações incluem o *depth testing* (teste com base na profundidade do fragmento) e o *blending* (combinação da cor do fragmento com a cor presente na mesma posição no *frame buffer*).

A quinta geração de GPUs (2004 ao início de 2005) inclui a placa GeForce 6800 da nVIDIA®. Nessa geração, a programabilidade foi também estendida, sendo que agora é possível a leitura de unidades de textura não apenas em programas de fragmentos, mas também em *shaders* de vértices. Além disso, o número máximo de instruções permitido nos *shaders* aumentou consideravelmente (de 512 para 65535). No entanto, o desempenho da aplicação fica comprometido se o programa de vértices ou fragmentos possuir um número muito grande de instruções. Outra melhoria foi o aumento da precisão em operações de *blending* e de filtragem de texturas, que agora trabalham com unidades de ponto flutuante de 16 bits, ao invés dos 8 bits suportados na geração anterior. Essas melhorias visam favorecer diversas técnicas de *rendering* em tempo real. Tais aplicações incluem além da própria visualização volumétrica, efeitos avançados como geração de sombras suaves, imagens com alta faixa dinâmica e sistemas de partículas.

2.3.2 Visualização Volumétrica em Tempo Real

Antes do advento das GPUs, os cálculos exigidos pelos algoritmos de visualização volumétrica em geral eram executados em sua totalidade na CPU. Isso dificultava muito a visualização interativa, visto que normalmente uma grande quantidade de dados é processada por esses algoritmos. Apesar do desenvolvimento de otimizações, apenas o avanço das tecnologias de hardware nos últimos anos possibilitou uma grande expansão na utilização dessas aplicações. Uma estratégia adotada foi o uso de arquiteturas paralelas, ou execução dos sistemas em vários computadores ligados em rede, forçando a implementação do programa de forma distribuída (ZUIDERVELD et al., 1996). Outra abordagem relevante é a utilização de hardware dedicado à visualização de volumes, como o *VolumePro.* O *VolumePro* foi o

primeiro chip para computadores pessoais que permitiu realizar o *rendering* de um volume em tempo real (PFISTER et al., 1999).

Nos últimos anos, pode-se notar que a estratégia mais explorada para melhorar a qualidade das imagens e prover interatividade nas aplicações de visualização consiste na exploração dos recursos das placas gráficas atuais. Diversas publicações recentes propõem a implementação de técnicas tradicionais de visualização na GPU (ENGEL et al., 2001; GELDER et al., 1996; KRÜGER et al., 2003; MEISSNER et al., 1999; ROETTGER et al., 2003; SWAN et al., 2003; WESTERMANN et al., 2001).

O suporte ao mapeamento de texturas em hardware foi um dos primeiros recursos a serem explorados (GELDER et al., 1996), se tornando uma solução bastante aceita para a visualização volumétrica direta em tempo real. A ideía é interpretar o volume de dados como sendo uma textura 3D discreta armazenada na GPU. Como um volume de dados não contêm uma geometria própria, e o *pipeline* gráfico atual se baseia na rasterização de primitivas geométricas para gerar imagens no *frame buffer*, é necessário gerar uma amostragem poligonal da textura para obter o *rendering* volumétrico. Normalmente são utilizados planos cuja orientação é paralela entre si e perpendiculares à direção de observação, como ilustrado na Figura 2.6. Esses planos são comumente denominados geometria de amostragem (*proxy geometry*). O volume armazenado como textura 3D é, então, mapeado para os planos de amostragem através de coordenadas de textura interpoladas na rasterização. Através dessa abordagem, a impressão visual transmitida por um volume semi-transparente pode ser simulada de forma eficiente em qualquer sistema gráfico capaz de suportar interpolação em texturas 3D (trilinear) e operações de *blending* a nível de fragmento.

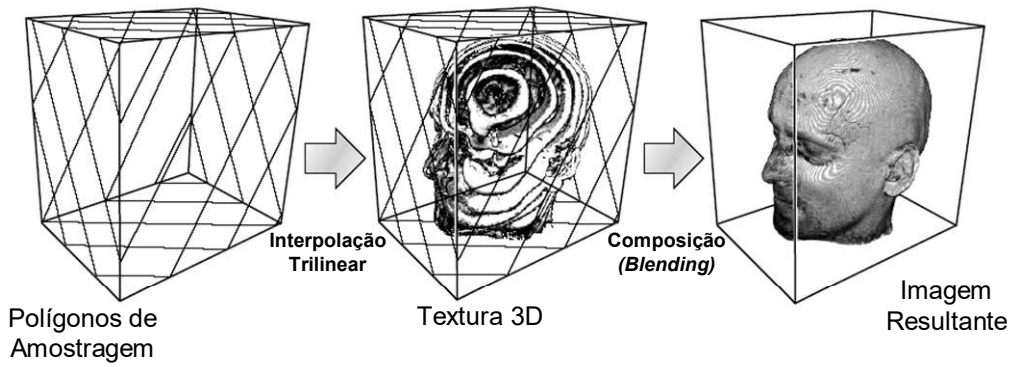


Figura 2.6: *Rendering* volumétrico baseado em texturas 3D. A geometria de amostragem consiste em planos ortogonais à direção de visualização (baseado em ENGEL et al., 2002).

Quando o hardware utilizado não suporta as texturas 3D (sendo capaz apenas de executar interpolações bilineares), ainda assim é possível a utilização de texturas 2D para armazenar o volume de dados (ENGEL et al., 2002). Isso é feito através da decomposição dos voxels em três pilhas de texturas 2D, uma para cada eixo principal de coordenadas(x, y, z). Para cada ponto de vista, é processada a pilha cujos planos encontram-se mais próximos (no sentido de orientação) da direção de observação. Caso fosse utilizada apenas uma pilha de fatias, em determinadas posições da câmera seria possível enxergar entre as fatias, o que prejudica consideravelmente a visualização. Apesar da réplica de dados na GPU, as operações envolvendo texturas 2D demonstram ter um desempenho superior às que usam texturas 3D,

visto que as últimas possuem uma complexidade consideravelmente maior em virtude da dimensão adicional. A Figura 2.7 ilustra a abordagem de *rendering* utilizando texturas 2D.

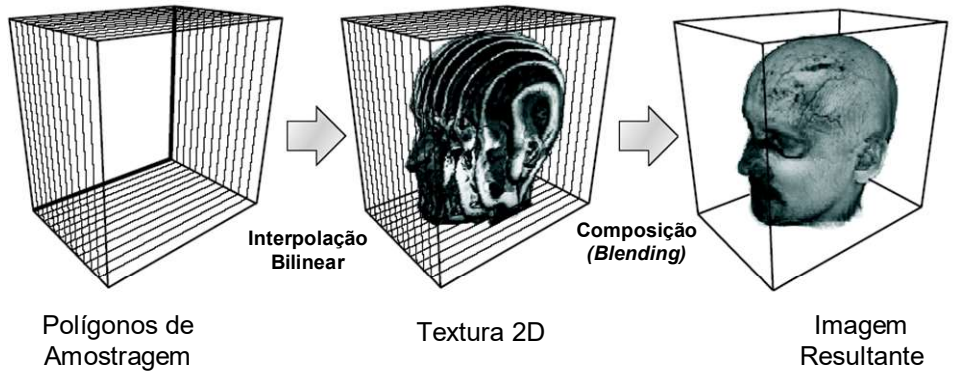


Figura 2.7: *Rendering* volumétrico baseado em texturas 2D. A geometria de amostragem consiste em planos alinhados com o objeto a ser visualizado (baseado em ENGEL et al., 2002).

A etapa de classificação executada através das FTs também pôde tirar proveito do suporte ao mapeamento de texturas pelo hardware. Antes da introdução das GPUs, as *lookup tables* que armazenavam as FTs residiam na memória da CPU, assim como o volume de dados. Cada vez que a FT sofria modificações, estas se refletiam na tabela e o volume precisava ser classificado e renderizado novamente. Quando executado na CPU, esse processo não costuma ser interativo, devido à complexidade das operações e quantidade de dados envolvidos. Isso dificulta ainda mais a tarefa de especificar FTs. Como a textura no hardware reside na área de memória da GPU, com os recursos de programabilidade do hardware se tornou possível utilizar texturas como meio de armazenamento das *lookup tables*. Assim, tanto a pré quanto a pós-classificação podem ser executadas no hardware com uma melhoria considerável no desempenho. Um meio de implementar essa classificação seria através do conceito de texturas dependentes introduzido na terceira geração de GPUs. Esse recurso permite aos fragmentos buscarem um valor em uma unidade de textura e utilizar esse valor como índice para o acesso a outra(s) unidade(s). Da mesma forma, um voxel utiliza seu valor escalar como índice de uma FT (representada por uma *lookup table*) para retornar seu valor de cor e opacidade.

Além da utilização de texturas para armazenar o volume, diversas técnicas de visualização direta e extração de isosuperfícies desenvolvidas recentemente tiram proveito do hardware a fim de prover interação em tempo real. Também é visada a melhoria na qualidade das imagens geradas através de visualização volumétrica. Isso é demonstrado pela implementação na GPU de novos efeitos para o rendering volumétrico, como cálculos de iluminação e tonalização (*shading*), sombras, filtros e mapeamentos de ambiente (*environment mapping*). Uma discussão detalhada sobre o assunto pode ser encontrada em (ENGEL et al., 2002).

2.4 Conclusão

A visualização volumétrica direta está consolidada como uma técnica flexível para o *rendering* de volumes de dados. O avanço e a flexibilidade de programação das arquiteturas gráficas atuais permitem que diversas técnicas de visualização sejam adaptadas ao hardware. Com isso, se tornou possível o *rendering* interativo de imagens volumétricas de altas

resoluções em computadores domésticos. O uso de texturas 3D e 2D como um meio de armazenamento dos voxels e também de FTs já está sedimentado como uma forma eficiente de visualização volumétrica direta (ENGEL et al., 2001). No entanto, um dos empecilhos em sua popularização é a especificação de FTs. No capítulo 3 são apresentadas técnicas que abordam esse problema, no intuito de auxiliar o usuário a encontrar a FT mais adequada para seu problema. Foi dada ênfase nos trabalhos mais relacionados com a proposta desta dissertação, abordando as vantagens e desvantagens de cada um.

3 ESPECIFICAÇÃO DE FUNÇÕES DE TRANSFERÊNCIA

Na seção 2.2 foram apresentados os métodos mais usuais para a geração de funções de transferência. Cabe ressaltar que os mesmos, normalmente devido à falta de conhecimento *a priori* dos dados sendo visualizados, acabam conduzindo o usuário a um processo de "tentativa e erro". Esse processo geralmente consome tempo e não é intuitivo, isso porque é observado que pequenas mudanças na FT resultam em drásticas mudanças na imagem resultante, como mostrado na Figura 3.1. Dessa forma, seria interessante que o método de especificação guiasse o usuário na escolha da melhor função para o volume de dados a ser visualizado. Nos últimos anos, têm sido desenvolvidas técnicas automáticas e semi-automáticas que têm como objetivo facilitar a tarefa da definição de FTs. De acordo com Kindlmann (KINDLMANN et al., 2002), é possível classificar essas abordagens em dirigidas aos dados (*data-driven*) ou dirigidas pela imagem (*image-driven*). Ambas serão discutidas a seguir. É importante mencionar que essa classificação não é restrita, uma vez que é possível desenvolver técnicas que misturem as características de ambas as classes.

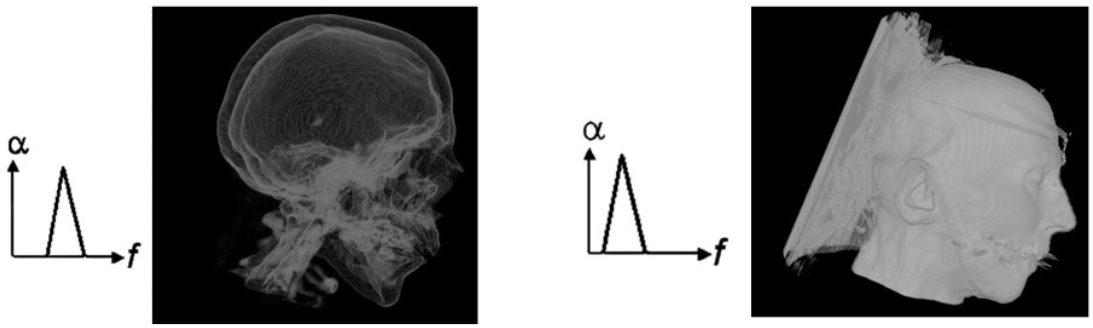


Figura 3.1: Mudanças pequenas na FT podem resultar em mudanças drásticas nas imagens resultantes.

3.1 Abordagens Dirigidas aos Dados

O princípio das técnicas para especificação de funções de transferência dirigidas aos dados consiste no fato de que a informação presente no volume de dados pode guiar o usuário na escolha da melhor função. Essa informação também é utilizada para limitar o espaço das funções de transferência a um conjunto mais relevante para a aplicação (KINDLMANN et al., 2002).

Muitos métodos dessa classe partem da premissa de que as funções de transferência mais adequadas são aquelas que realçam regiões de interesse dentro do volume. Normalmente é considerada região de interesse a fronteira entre dois materiais distintos. Dessa forma, o

problema consiste em detectar regiões de transição entre os materiais amostrados no volume, atribuindo valores de opacidade maiores para regiões de transição e menores para regiões homogêneas.

Seguindo essa idéia, o método semi-automático de Kindlmann (KINDLMANN et al., 1998) se baseia na análise de um histograma 3D que armazena, para um volume de dados, a correlação entre valor da amostra, a primeira e a segunda derivadas direcionais ao longo da direção do gradiente. De acordo com os fundamentos de processamento de imagens (GONZALEZ et al., 2000) , tanto os pontos de máximo da primeira derivada quanto os cruzamentos em zero da segunda indicam uma transição nos valores dos voxels. Essa transição pode ser interpretada como uma borda entre diferentes regiões homogêneas. Assim, do histograma 3D são extraídos *distance maps* (mapas de distância) que informam o quão próximo de uma região de transição um determinado voxel se encontra. De posse desses mapas, o usuário fornece uma função de ênfase nas bordas (*boundary emphasis function*), que atribui uma quantidade maior ou menor de opacidade para os voxels considerados em uma região de transição ou próximos a uma delas. É importante notar a diferença entre um método de *rendering* volumétrico direto, onde a FT ressalta as bordas entre os objetos, e uma abordagem de *rendering* de iso-superfícies. Apesar da semelhança quanto às regiões de interesse a serem visualizadas, o trabalho de Kindlmann (KINDLMANN et al., 1998) abordou essa questão através de uma comparação entre imagens geradas por ambos os métodos (Figura 3.2).

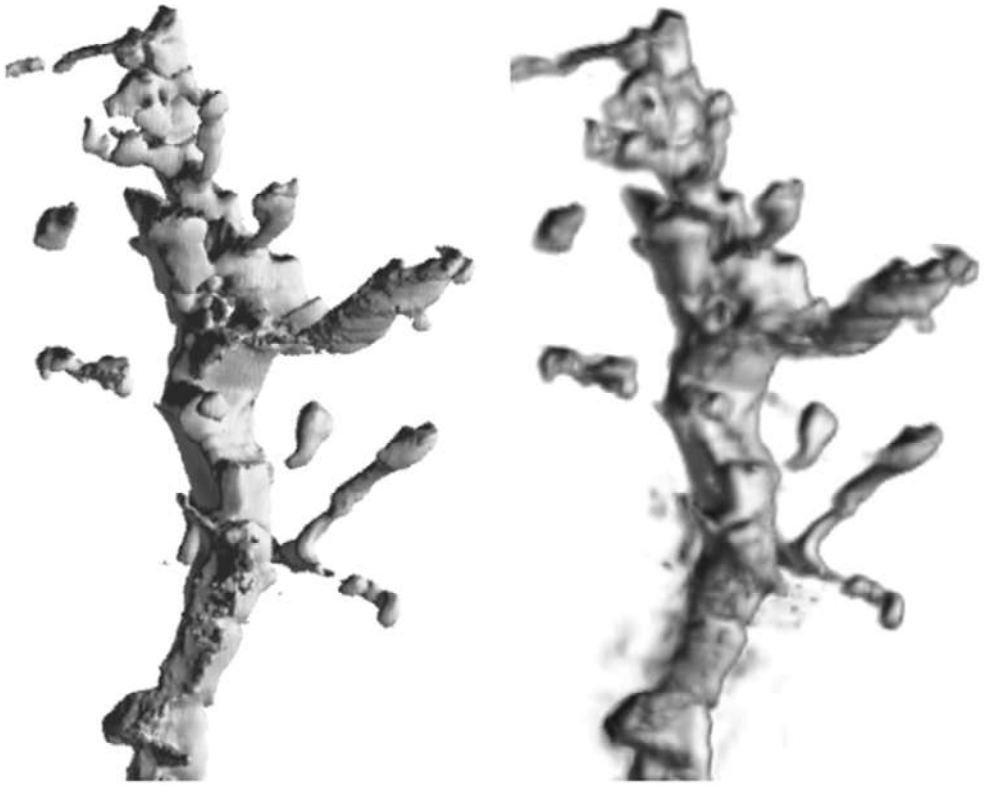


Figura 3.2: Tomografia de um dendrito renderizado a) através de extração de iso-superfície b) através de visualização volumétrica direta (extraído de KINDLMANN et al., 1998). Os parâmetros e iluminação e *shading* são os mesmos para ambas as imagens.

Com relação à Figura 3.2, pode-se notar que nas regiões ruidosas ou que não possuíam uma borda bem definida, o método de iso-superfícies deixou de considerar alguns voxels. Esses voxels poderiam ser artefatos gerados na aquisição, mas também poderiam fazer parte do dendrito. Na Figura 3.2b podemos observar a diferença no *rendering* direto de volume, que apresenta uma imagem mais borrada e com artefatos, geralmente provenientes da aquisição, porém com mais informação apesar de também estar exibindo a superfície externa do dendrito.

Outro método dirigido aos dados é o de Bajaj (BAJAJ et al., 1997), denominado *Contour Spectrum* e voltado à definição de iso-valores em malhas 3D não-estruturadas. Através da exploração de propriedades matemáticas da malha, algumas medidas importantes de uma iso-superfície podem ser computados de maneira eficiente. Tais medidas incluem a área da superfície, volume, média da magnitude do gradiente, bem como características topológicas. Essas medidas obtidas são exibidas graficamente na mesma interface onde é alterado o iso-valor e a imagem resultante é exibida. A Figura 3.3 mostra a interface do *Contour Spectrum*.

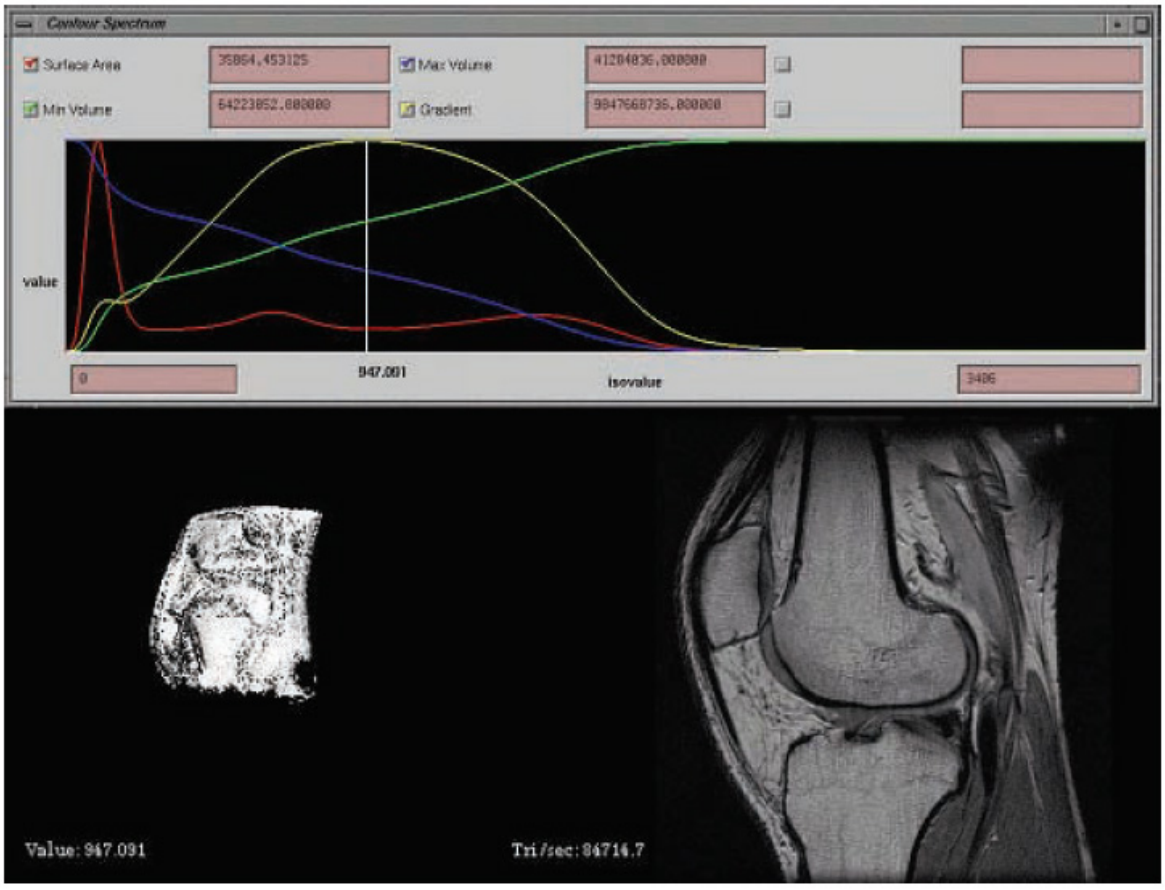


Figura 3.3: Interface do *Contour Spectrum* renderizando um volume obtido através de RM de um joelho. O gradiente aparece em amarelo no gráfico. Ajustando o iso-valor, tem-se o realce das bordas dos objetos (BAJAJ et al, 1997).

3.2 Abordagens Dirigidas pela Imagem

Nessa classe de algoritmos, normalmente são utilizadas imagens renderizadas a fim de explorar o espaço das funções de transferência. Suas técnicas são consideradas mais intuitivas, permitindo ao usuário escolher a melhor imagem renderizada, ao invés de alterar manualmente a função de transferência. No entanto, técnicas totalmente automáticas não são as mais adequadas, pois é necessário o controle do usuário no processo para refinar a separação entre as estruturas de interesse.

O método denominado *Design Galleries* proposto por Marks (MARKS et al., 1997) aborda o problema da especificação de parâmetros para aplicações de *rendering* em geral. Tais parâmetros incluem posição de luzes para o *rendering* de cenas 3D sintéticas, controle de movimento para animação de partículas, e também especficação de FTs para visualização volumétrica direta. Esse método parte da premissa de encontrar automaticamente o maior número de configurações de parâmetros possível, ficando a cargo do usuário a escolha da melhor função de transferência resultante dentre as configurações apresentadas. Para isso, é gerado aleatoriamente um grande número de FTs candidatas. Essas funções então passam por um processo de perturbação, também randômico, e uma seleção, que mantém os candidatos que tiverem uma variabilidade maior em relação aos outros. O sistema então exibe pequenos *thumbnails* 2D resultantes da aplicação dessas funções escolhidas. Essas imagens são dispostas, agrupadas de acordo com sua semelhança, em um arranjo denominado *Design Gallery*. O usuário, a seguir, navega nessas imagens, selecionando a que for mais adequada para a aplicação desejada. Devido ao grande número de funções geradas, é possível encontrar boas funções. No entanto, a desvantagem desse método é a não-intervenção do usuário na geração das funções de transferência iniciais, o que acaba tornando-o muito automático (PFISTER et al., 2001). Além disso, pela necessidade de geração de uma grande quantidade de *thumbnails,* o processo não é feito em tempo real, o que prejudica a interação. A Figura 3.4 mostra a interface do programa.

Figura 3.4: Interface do programa *Design Galleries* (MARKS et al., 1997).

Tzeng (TZENG et al., 2003) desenvolveu uma técnica dirigida à imagem, visando auxiliar tarefas de segmentação volumétrica em conjuntos de dados médicos. Nesse sistema, o usuário, através de uma interface gráfica, interage pintando regiões de interesse nas fatias do volume. Uma rede neural implementada como um programa de fragmento na GPU é treinada de modo a atribuir opacidade a essas regiões e assim ressaltá-las, fazendo uso de informações tais como magnitude do gradiente, vizinhança e posição espacial do voxel. Essa rede neural tem a função de "aprender" com as informações passadas pelo usuário, e produzir uma função que mapeia voxels em termos da incerteza de o mesmo pertencer a algum material de interesse. Essa incerteza pode ser mapeada para opacidade durante o *rendering*. A ferramenta também permite a remoção de estrutuas do volume através da pintura em fatias de amostra.

Apesar da utilização do hardware gráfico para os cálculos da rede neural e para o *rendering*, o algoritmo atingiu a taxa de 1,5 quadros/segundo. Apesar do baixo desempenho, a utilização do modelo de redes neurais facilita a paralelização do processamento, o que poderia acelerar o desempenho. A Figura 3.5 mostra alguns resultados obtidos com esse método.

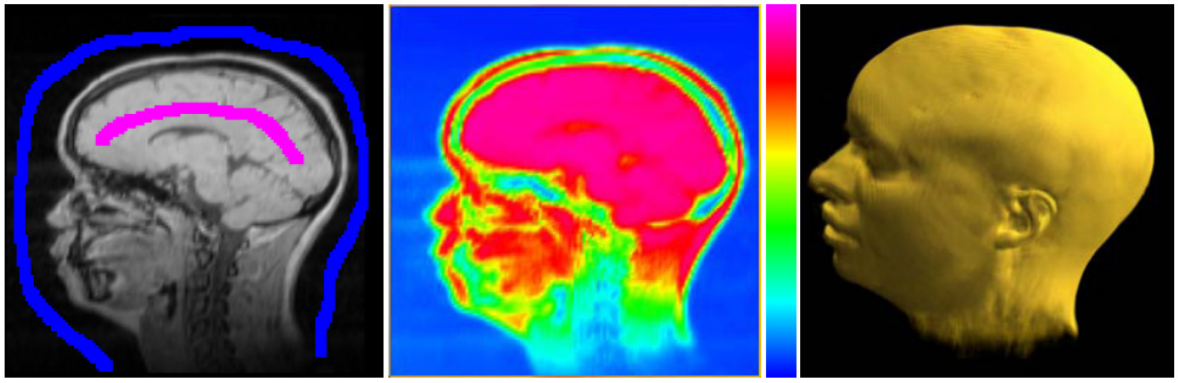


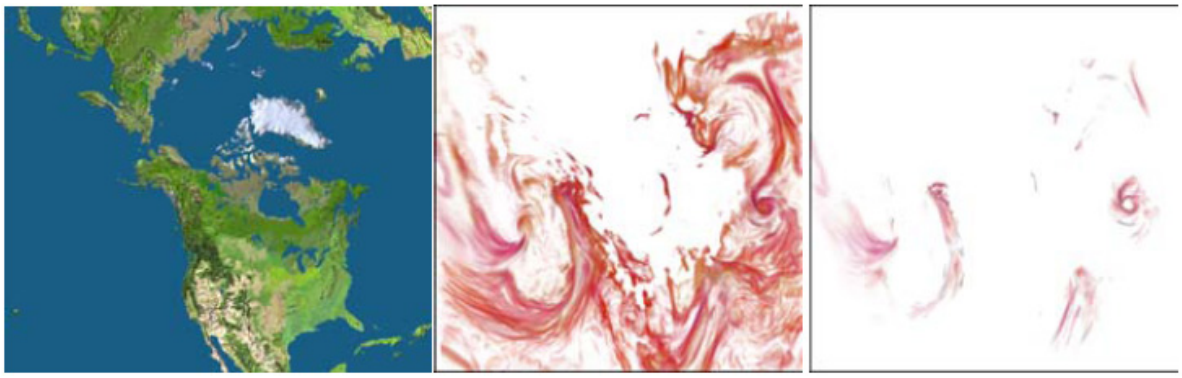
Figura 3.5: A imagem da esquerda mostra uma fatia pintada pelo usuário onde rosa representa o material de interesse e azul representa outros materiais. A imagem no centro mostra o resultado da classificação juntamente com a barra de cores associada. A imagem da direita é o resultado do *rendering* do volume classificado. (TZENG et al., 2003).

3.3 Especificação de FTs Multidimensionais

Funções de transferência mutidimensionais são as que possuem como domínio uma ou mais dimensões adicionais além do valor escalar do voxel, e podem mapear atributos visuais diferentes para cada combinação possível de suas dimensões de entrada. Um problema potencial da especificação de FTs em geral é quando um volume de dados apresenta diferentes materiais ou regiões de interesse que compartilham intervalos semelhantes de valores escalares. Normalmente conjuntos de dados médicos apresentam esse cenário, o que dificulta o realce de uma região em particular, visto que outras regiões com valores semelhantes também são afetadas pela FT em questão. As FTs multidimensionais surgiram com o intuito de resolver esse problema, provendo mais informação associada a um voxel, de modo a facilitar a diferenciação de estruturas de interesse do volume. Um exemplo de dimensão adicional é a magnitude do gradiente (LEVOY, 1988) que pode ser computada para cada voxel do volume e utilizada no domínio da FT. No entanto, a tarefa de especificar FTs unidimensionais, que consideram apenas o valor do voxel, já apresenta desafios, que se tornam ainda mais complexos na definição de FTs multidimensionais. Além disso, geralmente FTs são armazenadas em *lookup tables* que residem na memória de texturas da placa gráfica, e FTs multidimensionais requerem um espaço que cresce exponencialmente com o número de dimensões, e podem se tornar inviávies em dimensões muito grandes.

De modo a atacar essas limitações, Kniss (KNISS et al., 2003) propôs o uso de primitivas Gaussianas para representar FTs ao invés de *lookup tables*. Essas primitivas são avaliadas explicitamente utilizando o hardware gráfico programável, sem a necessidade de armazenar tabelas. Além da forma mais compacta de representação, FTs Gaussianas também permitem um cálculo mais preciso da integral de *rendering* volumétrico na GPU. Isso resulta em imagens de qualidade utilizando-se poucas fatias. Cada dimensão adicional do volume acarreta uma função diferente a ser avaliada para cada quadro. O método apresentou taxas de quadros de aproximadamente 3 quadros/segundo para funções com três dimensões de valores. Apesar da baixa taxa de quadros, cabe ressaltar que volumes de dados multidimensionais apresentam um complexidade exponencial com relação às dimensões em questão, o que torna

o processo muito custoso computacionalmente. A Figura 3.6 mostra um resultado obtido por esse método.



(a) (b) (c)

Figura 3.6: Resultados obtidos em uma simulação meteorológica usando o método de Kniss. O conjunto de dados possui 4 dimensões ou tipos de dados: temperatura, umidade, velocidade do vento e gradiente. (a) a região ou domínio da simulação. (b) exibição apenas da movimentação do vento na simulação. (c) seleção das regiões cuja velocidade do vento ultrapassou um determinado limiar (KNISS et al., 2003).

Outra abordagem que trata da especificação de FTs multimensionais é uma extensão ao trabalho de Kindlmann (KINDLMANN et al., 1998), desenvolvida por Kniss (KNISS et al., 2001). Essa extensão consiste numa interface para a especificação de FTs multidimensionais através da manipulação de objetos gráficos. Esses objetos permitem a atribuição de cor e opacidade tanto no domínio da FT (utilizando um gráfico) quanto no domínio espacial (com um lápis que aponta regiões de interesse do volume). Uma das dimensões adicionais é a magnitude do gradiente, que foi utilizada também para o cálculo de modelos de iluminação e tonalização (como aproximação do vetor normal à superfície) implementados nessa abordagem. A Figura 3.7 ilustra a interface do método de Kniss. No gráfico da figura 3.7 (parte inferior), a dimensão horizontal é o valor do voxel e a vertical é a magnitude do gradiente. Os triângulos invertidos (exibidos nas cores verde e vermelho) atribuem opacidade e cor para as regiões do gráfico onde estão posicionados. O lápis próximo à imagem do volume renderizado serve para apontar estruturas de interesse dentro do mesmo, cujas faixas de valores apontados são exibidos no gráfico em tempo real.

3.4 Conclusão

Pode-se notar a complexidade da tarefa de especificar FTs analisando os diversos métodos existentes na literatura. É importante observar que muitas vezes o usuário, apesar de possuir conhecimento sobre os dados em uso, encontra dificuldades para especificar um mapeamento aceitável quando na edição manual de uma FT. Dessa forma, uma das questões que surgem é qual seria o nível de interação adequado. Esse tema é abordado no capítulo 4, onde são descritos os passos e a implementação da ferramenta proposta.

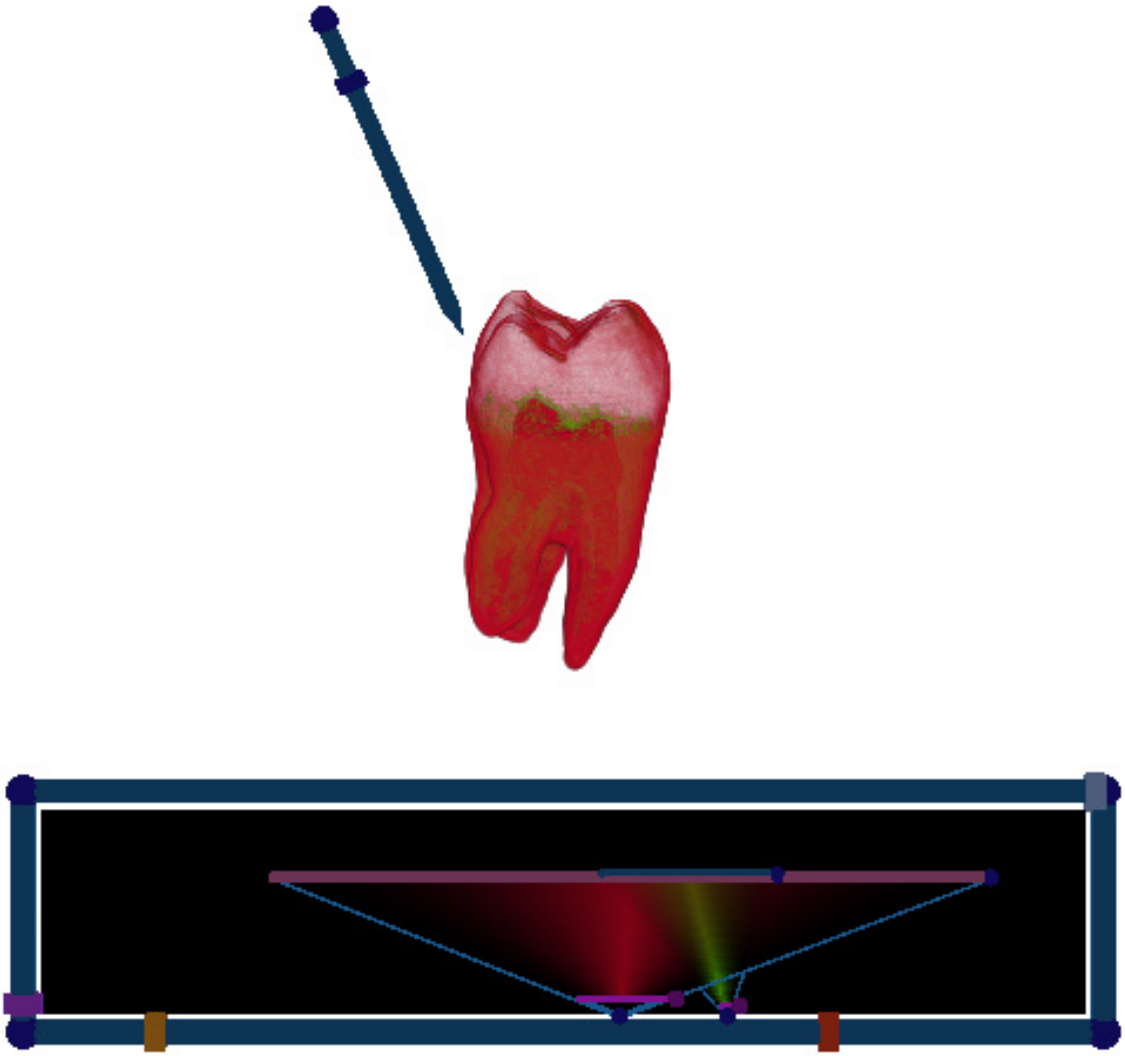


Figura 3.7: Interface do método de Kniss (KNISS et al., 2001) renderizando um volume de um dente obtido através de TC.

4 ESPECIFICAÇÃO DE FUNÇÕES DE TRANSFERÊNCIA EM DOIS NÍVEIS DE INTERAÇÃO

4.1 Visão Geral da Ferramenta

Em interfaces tradicionais de visualização volumétrica, é necessário que o usuário possua um certo grau de conhecimento sobre o conjunto de dados sendo visualizado para que seja obtida uma boa FT. Além disso, cada novo volume apresenta novos desafios e, dessa forma, o usuário poderia perder muito tempo definindo mapeamentos aceitáveis (HÖNIGMANN et al., 2003). Com relação à abordagem proposta, é explorada a possibilidade dos usuários não possuírem um conhecimento *a priori* sobre o volume. Para isso, foi adotada uma abordagem de geração de FTs iniciais de forma automática. No entanto, se o usuário possuir um conhecimento avançado é possível tirar proveito do mesmo, através de interações e refinamentos das funções candidatas apresentadas incialmente. Dessa forma, a ferramenta proposta nesta dissertação possui dois níveis de interação com o usuário de modo a englobar diferentes tipos de aplicações de interesse.

Dentre as várias técnicas descritas na literatura, foram considerados mais relevantes ao propósito desta dissertação o trabalho de Kindlmann e Durkin (KINDLMANN et al., 1998) e o método *Design Galleries* (MARKS et al., 1997). Com relação ao último, a disposição de várias FTs candidatas em um arranjo de *thumbnails* pode permitir ao usuário alguma identificação imediata sobre os dados, antes da interação. Um problema que afeta essa abordagem é que normalmente os dados volumétricos apresentam algum material de interesse envolto em outros materiais menos importantes no momento. No caso da atribuição de opacidade a esses voxels que não os de interesse, os mesmos acabam ocludindo estruturas internas importantes. De modo a solucionar esse problema, poderiam ser consideradas como regiões de interesse as fronteiras entre diferentes regiões homogêneas do volume (KINDLMANN et al., 1998).

Ainda com relação ao *Design Galleries*, as FTs apresentadas ao usuário na forma de *thumbnails* 2D são o resultado de processos de geração randômica e dispersão das FTs candidatas com base em suas similaridades. Dessa forma, a dispersão tem como objetivo substituir as FTs candidatas muito semelhantes por novas, geradas também aleatoriamente, a fim de prover uma maior variabilidade dos resultados. Por fim, o método organiza os *thumbnails* segundo suas semelhanças (por exemplo, os mais opacos em um grupo), e o usuário escolhe uma imagem de interesse a ser exibida com detalhe. Apesar de construir um arranjo com um número muito grande de candidatos, pode-se notar a automação excessiva desse método quando da especificação das FTs. Além disso, devido às limitações do hardware gráfico da época, a geração dos *thumbnails* para cada conjunto de dados diferente era feita de forma *off-line*.

A abordagem proposta visa explorar algumas dessas limitações, concentrando-se mais na redução do espaço de possíveis FTs para um mais razoável aos propósitos do usuário. Para isso, é também explorada a idéia de construir um arranjo de *thumbnails* representando FTs candidatas. No entanto, uma decisão importante na implementação da ferramenta consiste na apresentação de um conjunto relativamente pequeno de *thumbnails* iniciais. Essa redução no escopo das FTs candidatas visa a geração de um conjunto mais adequado, realçando regiões de interesse e provendo assim mais uma visualização mais informativa do volume.

Para realçar essas regiões, é necessário primeiramente definir o que pode ser considerado como uma região de interesse do volume. Na ferramenta proposta, são considerados voxels de interesse aqueles que se encontram próximos ou em uma região de transição entre materiais homogêneos do volume (KINDLMANN et al., 1998). É proposto o algoritmo de dispersão baseada nas bordas, cujo objetivo é gerar de forma automática FTs que possuam a capacidade de realçar bordas do volume. Além disso, é explorado o *rendering* em tempo real através da utilização dos recursos do hardware gráfico moderno. Isso é incluído num primeiro nível de interação, onde as FTs candidatas são aplicadas ao volume e exibidas na forma de *thumbnails* 3D. Esses *thumbnails* podem ser visualizados e manipulados de forma interativa pelo usuário. O segundo nível de interação visa atacar as limitações da automação existente no primeiro nível, provendo um maior controle ao usuário no refinamento de uma FT de interesse. Nesse nível é possível visualizar e interagir com o volume em detalhes, não mais na forma de *thumbnails,* e também alterar a FT de cor e opacidade, obtendo-se um novo *rendering* do volume em tempo real.

O processo global de classificação e visualização na técnica proposta é mostrado na Figura 4.1. Basicamente, um volume de dados é obtido através de um conjunto de imagens 2D (ou fatias) agrupadas umas sobre as outras. A partir desse volume é extraído um histograma 3D, que guiará o processo de criação e dispersão das FTs candidatas (denominadas vetores FT na Figura 4.1). Essas funções, então, podem ser aplicadas ao volume tanto no primeiro quanto no segundo nível de interação. Os conceitos e a implementação de cada uma das etapas mostradas na Figura 4.1 são discutidos em detalhe nas próximas seções.

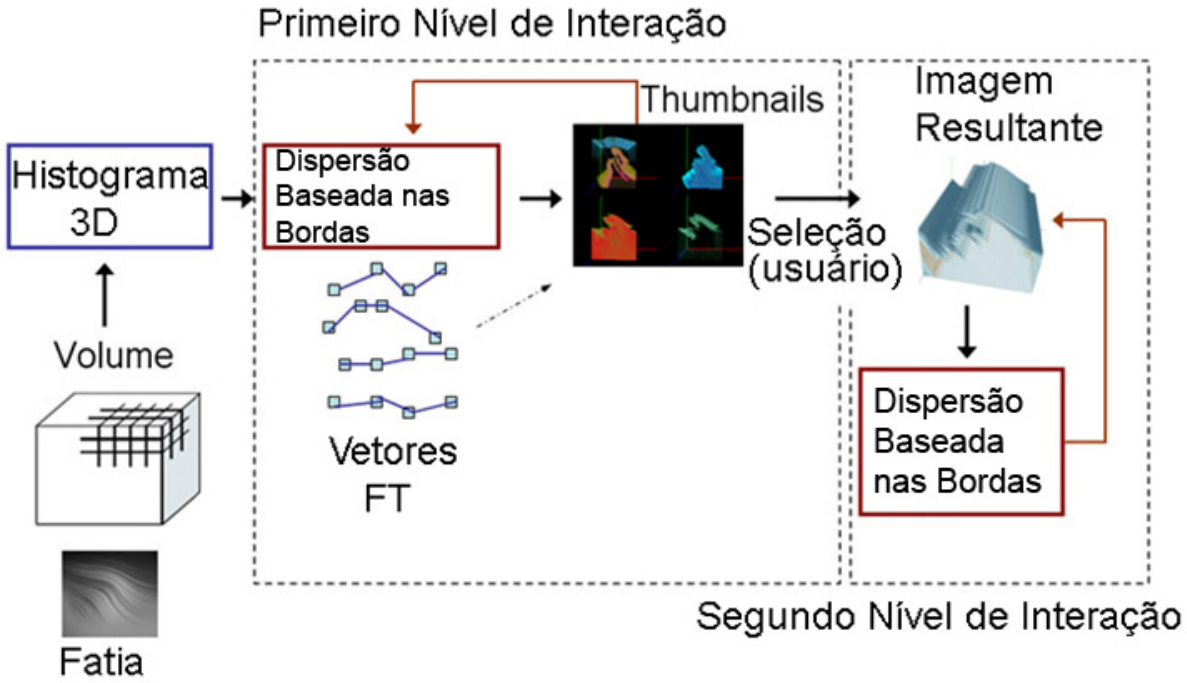


Figura 4.1: O método interativo de especificação de FTs proposto nesse trabalho. A dispersão baseada nas bordas é usada para gerar FTs resultando diferentes *thumbnails*, e, no segundo nível de interação, produzindo diferentes imagens volumétricas.

4.2 Preparação dos Dados para as FTs Inicias

Um dos objetivos desse trabalho é evitar a abordagem de tentativa e erro quando da especificação de FTs. Dessa forma, a abordagem proposta começa com a geração automática das FTs candidatas iniciais, seguindo o *Design Galleries* (MARKS et al., 1997). Nessa abordagem, isso é feito através da geração randômica e dispersão de uma grande quantidade de vetores de entrada (*input vectors*), os quais representam os pontos de controle de FTs de opacidade (um exemplo é mostrado na Figura 2.3), e são utilizadas para exibir o volume na forma de *thumbnails*. Antes da execução do *rendering*, um processo de dispersão (*evolution-disperse)* analisa as semelhanças nos vetores de entrada e também em um conjunto arbitrário de pixels vizinhos das imagens exibidas. A partir dessa análise, é gerada uma nova população de FTs candidatas, que substitui a anterior caso seja constatada uma maior variedade dos *thumbnails* resultantes.

A premissa do método *Design Galleries* é mostrar o maior número de possibilidades geradas aleatoriamente de modo que o usuário escolha a mais adequada. Já na abordagem proposta nessa dissertação, a idéia é encontrar diretamente um conjunto menor de combinações de FTs que realcem as bordas entre diferentes regiões do volume. Isso é feito com o intuito de reduzir o vasto espaço de FTs possíveis para um mais razoável. Para isso, antes da geração das FTs inciais e do primeiro nível de interação com o usuário, é necessário um modo de detectar a informação de interesse presente no volume. Na ferramenta proposta, essa informação consiste em saber se um voxel está em uma zona de transição ou se encontra próximo de uma. Dessa forma, é possível especificar FTs que atribuam uma maior opacidade para os voxels que possuam as características detectadas. É necessário fazer uma distinção entre FTs que realcem as bordas e as técnicas de segmentação volumétrica e detecção de bordas em geral. Técnicas de segmentação (tanto em imagens 2D quanto 3D) geralmente utilizam a posição espacial dos elementos da imagem nos processos que detectam os objetos de interesse. Além disso, o usuário normalmente interage com a técnica informando quais as regiões a segmentar (isolar) ou através de ferramentas de recorte. Já na especificação de FTs, não é necessária a posição do voxel no volume, pois é observado que o domínio de uma FT 1D não inclui posição, somente o valor escalar do voxel. Assim, a atribuição de opacidade a um escalar causa mudanças que se refletem em todos os voxels que possuam esse valor.

Para detectar os voxels pertencentes a regiões de transição do volume e então utilizar a informação para guiar a geração das FTs iniciais, foi implementada a técnica de histograma 3D proposta por (KINDLMANN et al., 1998). Armazenando o relacionamento entre o valor de um voxel e suas primeira e segunda derivadas ao longo da direção do gradiente, de modo que a posição espacial do voxel não é necessária.

Sabe-se que a primeira (f') e segunda derivadas (f'') em uma imagem codificam a taxa em que os valores estão mudando em uma dada direção. Levoy (LEVOY, 1988) observou que a magnitude do gradiente poderia ser usada como entrada adicional para a especificação de uma FT, de modo a realçar as bordas do volume. Kindlmann (KINDLMANN et al., 1998) estendeu essa idéia ao observar que tanto os pontos de máximo (ou pontos de inflexão) do gráfico de f', assim como os cruzamentos em zero de f'' (ao longo da direção do gradiente) indicam uma mudança nos valores escalares. Tal mudança pode ser entendida como uma fronteira entre regiões homogêneas. Esse conceito é ilustrado na Figura 4.2.

O histograma 3D é uma forma compacta de relacionar os valores de f' e f'' medidos para todo o volume da dados. Essa relação se dá na organização das dimensões (ou eixos) do histograma. Uma dimensão é o valor do voxel. As outras são a f' (ou magnitude do gradiente)

e a *f''* derivadas, ambas medidas para cada voxel do volume. Cada elemento do histograma informa o número de ocorrências de uma combinação particular de valor escalar, *f'* e *f''* no volume.

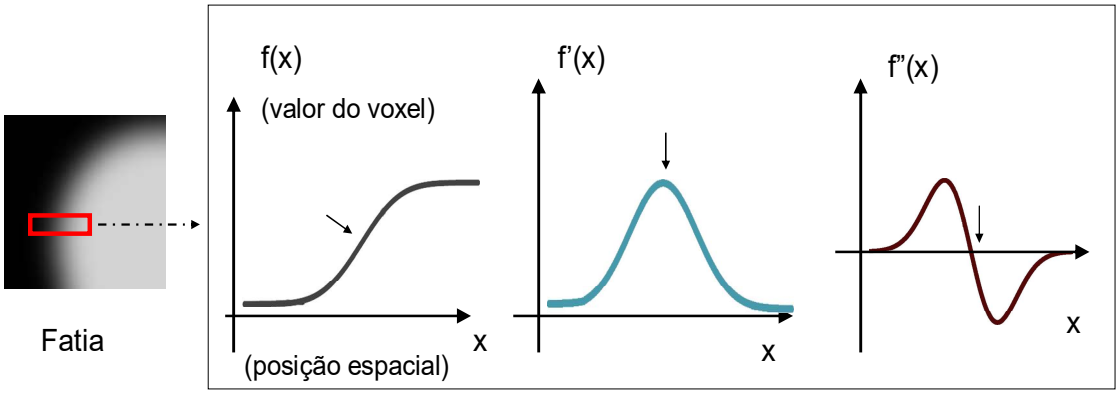


Figura 4.2: Análise de uma borda típica em uma imagem 2D(fatia). O trecho marcado em vermelho na fatia é representado como uma função contínua *f(x)*, juntamente com a primeira (*f'(x)*) e a segunda (*f''(x)*) derivadas de *f(x)*. As setas indicam nos gráficos o que seria o centro da zona de transição.

Após a leitura do volume de dados proveniente de uma pilha de fatias 2D, ou de uma imagem 3D, é feito o cálculo do histograma 3D. A fim de reduzir o espaço necessário em memória para o armazenamento do mesmo, ao invés de guardar todos os valores possíveis de *f'* e *f''*, é feita a divisão dos valores em intervalos denominados *bins*. Assim, um *bin* é o menor elemento do histograma, e contabiliza as ocorrências de um valor de voxel com um pequeno intervalo de valores de *f'* e *f''* presentes no volume. O cálculo do histograma 3D necessita de duas passadas no volume. A primeira delas é feita para medir os valores máximos e mínimos de *f'* e *f''* e assim fazer a divisão do histograma em *bins*, que são preenchidos na segunda passada. A implementação desses procedimentos segue o seguinte algoritmo:

- Percorrer o volume calculando *f'* e *f''* para cada voxel (primeira passada);
- Armazenar o valor máximo de *f'* e *f''*, e o mínimo de *f''* (assume-se zero o mínimo de *f'*; já *f''* pode ter valores negativos);
- Dividir o intervalo de valores possíveis de *f'* e *f''* em *x bins* (assume-se *x* igual ao número de valores escalares possíveis do volume);
- Inicializar todos os *bins* do histograma com o valor zero;
- Percorrer o volume (segunda passada). Para cada voxel:
    - Calcular f' e f'';
    - Encontrar o *bin* que contém o valor do voxel e o intervalo onde se encontram os valores medidos;
    - Incrementar o valor do *bin* correspondente.

Para o cálculo de *f'* (magnitude do gradiente), é utilizado o método de diferenças centrais aplicada a imagens 3D. Para cada voxel, é calculada a diferença entre seus vizinhos imediatos, nas direções x, y e z. O resultado é um vetor (gradiente) contendo essas três componentes, e cujo módulo representa a *f'*. Para o cálculo de *f''*, é utilizado o operador Laplaciano, que comumente representa uma aproximação da segunda derivada em imagens 2D. Para a convolução do operador Laplaciano no volume, é utilizada uma máscara de convolução

3x3x3. Esses métodos foram escolhidos em virtude de sua simplicidade de implementação, ainda que precisos o bastante para detectar diversas configurações diferentes de bordas. A Figura 4.3 ilustra a estrutura do histograma 3D.

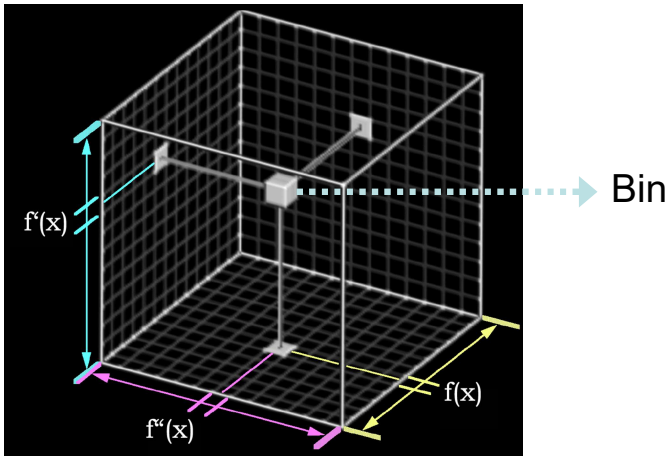


Figura 4.3: Histograma 3D exibindo um *bin* específico. Cada dimensão (*f(x)*, *f'(x)* e *f''(x)*) é dividida em pequenos intervalos de valores. O *bin* armazena o número de vezes que uma determinada combinação desses intervalos aparece no volume.

4.3 Exibição dos *Scatterplots* do Histograma 3D

Após o cálculo do histograma 3D, é necessária uma forma de extrair as informações relevantes que possam auxiliar o usuário ou o sistema a encontrar uma FT adequada. Kindlmann (KINDLMANN et al., 1998) observou que o *rendering* de projeções do histograma 3D pode informar em que faixas de valores de dados se encontram as bordas entre diferentes regiões do volume. Essas projeções são denominadas *scatterplots* do histograma (Figura 4.4), e consistem em gráficos 2D cuja dimensão horizontal é o valor do voxel e as dimensões (ou eixos) verticais são os intervalos de *f'* e *f''*, respectivamente. A quantidade de ocorrências no volume de uma combinação particular de valor de voxel, *f'* e *f''* é armazenada no histograma 3D.

Assim, um valor de voxel pode ocorrer no volume com diferentes medidas de primeira e segunda derivadas. A solução adotada por Kindlmann (KINDLMANN et al., 1998) para a exibição desses *scatterplots* é o mapeamento do número de ocorrências do histograma para níveis de cinza. Quanto menor o número de ocorrências (valor armazenado em um *bin*), mais baixo (escuro) é o nível de cinza, e vice-versa. A exibição dos *scatterplots* tem o objetivo de prover mais informação e auxiliar o usuário a localizar diferentes características no volume. É possível identificar quais os valores de voxels (e quantos deles) se encontram em uma região homogênea ou em uma região de transição (borda). Além disso, exibir diretamente o histograma 3D (como um volume de dados) em níveis de cinza dificultaria sua compreensão, visto que o mesmo normalmente possui uma grande densidade de elementos, e grande parte sofreria oclusão.

Um problema existente no mapeamento de ocorrências do histograma 3D para níveis de cinza é a quantização que ocorre no processo. Enquanto geralmente são utilizados apenas 256 níveis de cinza, no mesmo histograma 3D podem ocorrer *bins* com valores como 10, 100 ou

100.000. Assim, esse mapeamento apresenta uma quantização inerente. A normalização de um valor, que consiste em sua divisão pelo maior valor encontrado no histograma, pode simplesmente descartá-lo do *scatterplot*. Dessa forma, somente os valores mais freqüentes no volume apareceriam no gráfico, e os menos freqüentes seriam descartados (atribuídos de valor zero).

Uma solução adotada para contornar essa limitação é mapear o número de ocorrências para cores ao invés de níveis de cinza, a fim de aumentar o intervalo de valores de destino e facilitar a visualização dos *scatterplots*. No entanto, um problema que surge é a escolha do espaço de cores a ser utilizado no mapeamento. O espaço RGB (*red*, *green*, *blue*), apesar de sua larga utilização em aplicações gráficas, não apresenta uma variação linear de tonalidades, ou seja, é complexa a variação do tom de uma mesma cor (mais escuro para mais claro). Em vista disso, foi escolhido o espaço de cores HSV (*Hue*, *Saturation*, *Value;* ou Matiz, Saturação e Valor). Ao invés de representar o espectro cromático através de combinações das cores básicas vermelho, verde e azul, o espaço HSV utiliza o matiz para descrever a tonalidade da cor. A saturação indica o quanto a cor predomina, e o valor altera o brilho ou intensidade do tom especificado pelo matiz.

Para o mapeamento de ocorrências do histograma para cores nos *scatterplots*, é escolhida uma variação de amarelo (poucas ocorrências) para vermelho (várias ocorrências), e a cor de fundo do gráfico é preto (significando a ausência de ocorrências). São utilizados valores de cor do espaço HSV para preencher uma tabela, onde o matiz varia de amarelo a vermelho e mantidos constantes os valores de saturação e intensidade. Finalmente, o número de ocorrências no histograma é o índice para acessar essa tabela e retornar a cor a ser desenhada no gráfico. Essa cor é então convertida para o espaço RGB antes da exibição, visto que o pipeline gráfico tradicional trabalha apenas com esse espaço de cores. A Figura 4.4 ilustra a exibição dos *scatterplots* com base em dois volumes, um sintético e outro obtido através de TC.

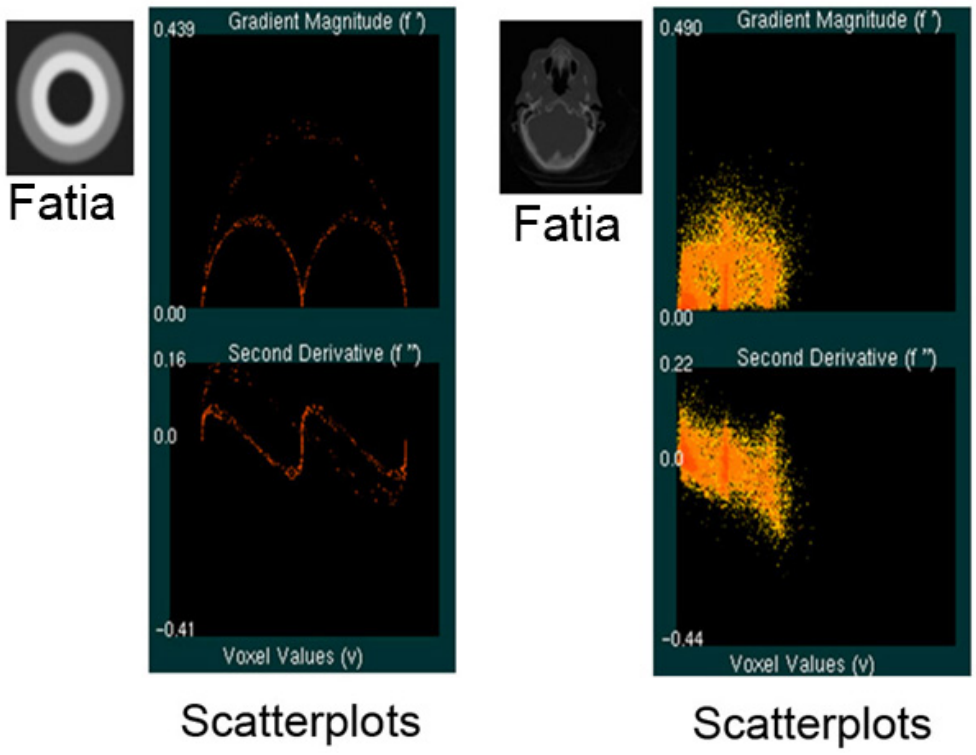


Figura 4.4: Projeções do histograma 3D: (a) volume sintético; (b) volume "UNC CT Head". Em ambos exemplos é possível identificar regiões de transição através dos pontos de máximo da magnitude do gradiente (*f'*) e os cruzamentos em zero de *f''*.

4.4 FTs Iniciais e Primeiro Nível de Interação

Como observado nos *scatterplots* mostrados na Figura 4.4, cada valor de voxel geralmente possui várias medidas diferentes de *f'* e *f''*. No entanto, para guiar a especificação das FTs apresentadas no primeiro nível de interação da ferramenta, é necessário associar apenas uma medida de *f'* e *f''* para cada valor escalar de voxel *(v)*. As médias de *f'* e *f''* são calculadas para cada valor de *v* e moduladas pelo total de ocorrências de *v* contidas no histograma 3D. Nesse ponto são definidas duas funções importantes para a geração das FTs iniciais. Denomina-se *g(v)* a média de *f'* associadas ao valor *v*, e *h(v),* a média de *f''*. Essas funções servem como indicadores do escore de cada valor de voxel *v*. Assim, os valores altos de *g(v)* (assume-se apenas valores positivos para *g(v)*) indicam os pontos de máximo de *f'*, ou seja, o voxel *v* potencialmente pertence a uma região de transição no volume. Da mesma forma, valores de *h(v)* próximos de zero (essa função pode resultar valores negativos) também indicam que *v* pode pertencer a uma borda.

O resultado de *g(v)* e *h(v)* extraído do histograma 3D para todos os voxels *v* do volume é armazenado em duas tabelas cujo número de elementos é igual ao número de possíveis valores de *v.* Cada tabela guarda o valor *v* e os valores de *g(v)* e *h(v)* associados a *v*, respectivamente. Após o preenchimento da tabelas, é feita uma ordenação considerando a ordem decrescente de *g(v).* Para o caso de *h(v)* a ordenação é feita considerando a proximidade de zero do valor da função, ou seja, os primeiros elementos são os mais próximos de zero, tanto negativos quanto positivos. As tabelas mencionadas são utilizadas para guiar o método de dispersão baseada nas bordas (*edge-based dispersion*), responsável pela geração das FTs iniciais mostradas como *thumbnails* 3D. Esse processo é ilustrado na Figura 4.5 e descrito em detalhes na próxima seção.

4.4.1 Dispersão Baseada nas Bordas

A técnica de dispersão baseada nas bordas é proposta para obter as FTs iniciais, onde os valores *v* com maiores escores ganharão mais opacidade nas FTs. No método *Design Galleries* (MARKS et al., 1997), as FTs são parametrizadas como vetores contendo pontos de controle, onde o índice do vetor corresponde a *v,* e os valores significam a opacidade atribuída a *v*. A dispersão randômica das opacidades nesse método é feita a fim de prover uma maior variação das FTs e dos *thumbnails* resultantes.

Na abordagem aqui proposta também são utilizados vetores para representar as FTs, os denominados vetores FT. Cada vetor FT possui oito posições, apesar de menos posições se mostrarem eficazes na maioria dos casos. Na dispersão baseada nas bordas, são utilizadas as informações das tabelas de *g(v)* e *h(v)* para preencher as posições dos vetores FT. Além disso, é também utilizada uma função definida por Kindlmann (KINDLMANN et al., 1998) para a combinação das informações de *g(v)* e *h(v)*, que é denominada *k(v, thickness)* e definida através da fórmula:

$$k(v, thickness) = \frac{h(v)^2 * thickness}{g(v)} \qquad \text{Equação (4.1)}$$

onde *thickness* é definida como uma medida relativa de largura da borda que se quer realçar. Quanto mais próximo de zero for o resultado de *k(v, thickness)*, mais próximo de uma borda o voxel *v* vai estar posicionado. Tanto valores baixos (cruzamentos em zero) de *h(v)* quanto valores elevados (máximos) de *g(v)* favorecem um resultado próximo de zero da função *k(v, thickness)*. O parâmetro *thickness* então modula a probabilidade de *k(v, thickness)* ser próximo de zero e, portanto, o voxel *v* fazer parte de uma região de transição. Na ferramenta proposta, o valor de *thickness* pode ser tanto especificado pelo usuário quanto randomicamente gerado. Os valores de *k(v, thickness)* associados a cada voxel também são ordenados (seguindo o mesmo critério de *h(v)*) e armazenados em uma tabela (K).

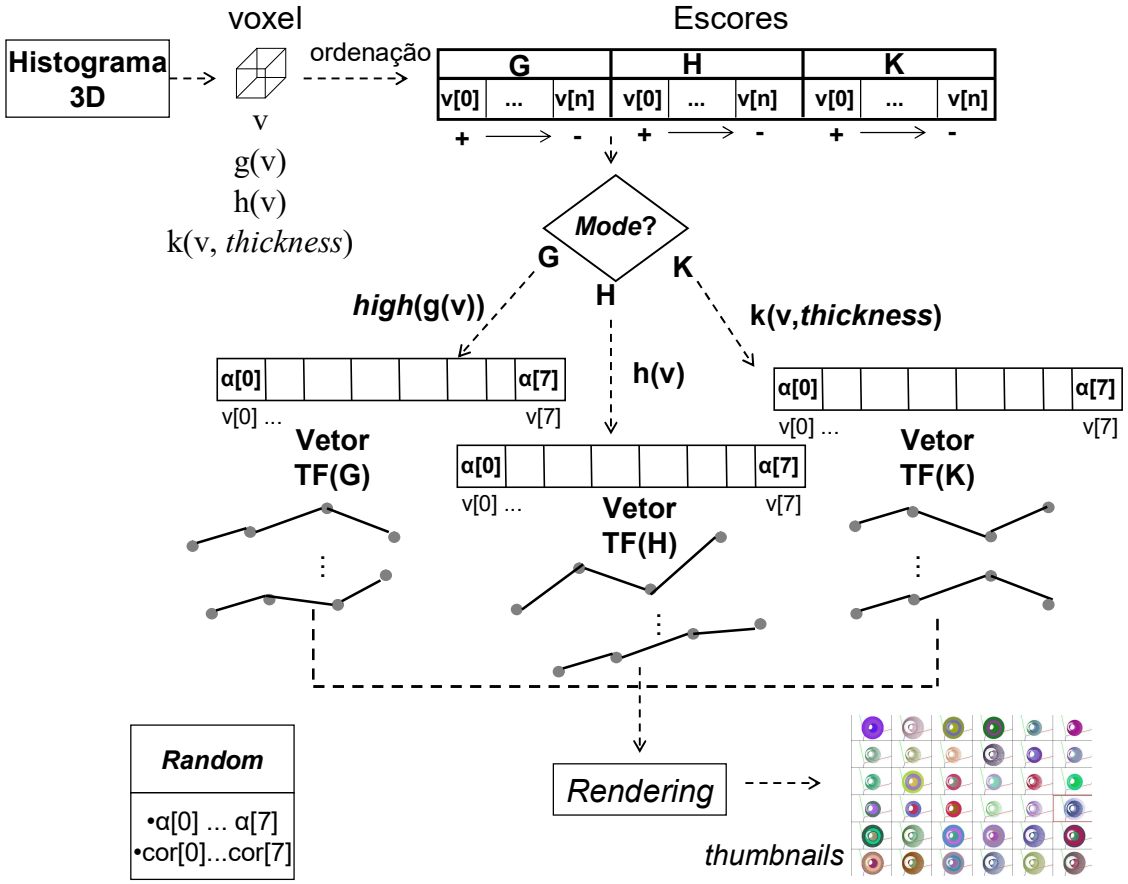


Figura 4.5: Sumário do algoritmo de dispersão baseada nas bordas (*edge-based dispersion*). As informações extraídas do histograma 3D guiam o processo de geração dos vetores FT.

Assim, apenas as oito primeiras posições das tabelas (as que possuem os melhores escores) são utilizadas nos vetores FT e, dessa forma, têm chance de ganhar um valor de opacidade diferente de zero nas FTs iniciais. A variável *mode* mostrada na Figura 4.5 decide qual tabela (F, G ou K) irá preencher um dos vetores FT. A função *high(g(v))* seleciona os valores de *v* que possuem os escores mais altos com respeito a *g(v)*. Nesse momento, os

vetores FT possuem em cada posição um valor de *v* que potencialmente pertence a uma região de borda do volume. A opacidade e a cor para cada posição são então atribuídas de forma aleatória, sendo que é permitido ao usuário arbitrar o valor máximo possível de opacidade. Cada vetor FT é, então, interpolado linearmente, resultando em uma *lookup table* (FT) completa, com a quantidade de valores de entrada igual ao número de diferentes valores escalares do volume. As *lookup tables* resultantes são, então, utilizadas para classificar o volume e renderizar um *thumbnail* individual. Na ferramenta desenvolvida, são exibidos simultaneamente 36 *thumbnails* dispostos em uma grade 6x6, como ilustra a Figura 4.6. Dessa forma, cada tabela (F, G e K) é utilizada para geração de 12 vetores FT, ou duas linhas de 6 *thumbnails* agrupados. Apesar da opacidade e cor serem atribuídas de forma aleatória para cada ponto de controle, a dispersão não muda os valores de voxels presentes nos vetores FT.

4.4.2 Primeiro Nível de Interação

O conjunto inicial de FTs é exibido como *thumbnails* 3D em tempo real utilizando os recursos do *hardware* gráfico programável. Cada *thumbnail* é exibido como uma *viewport* diferente organizada na mesma janela, como ilustrado na Figura 4.6, onde foi utilizado um volume de dados sintético de 64^3 voxels contendo um cilindro dentro de outro. Uma fatia de amostra desse volume de dados é ilustrada na Figura 4.4a.

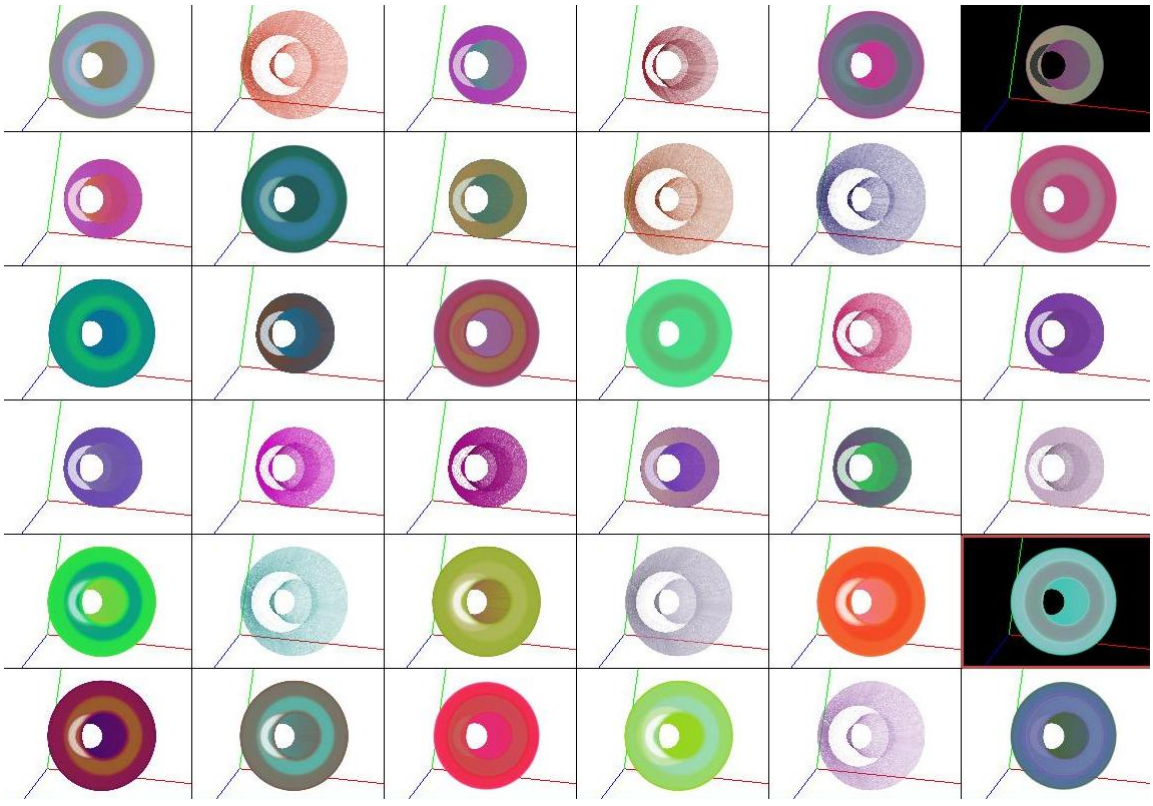


Figura 4.6: Conjunto de FTs iniciais para o mesmo volume de dados sintético, exibidas como *thumbnails* 3D. Aqueles mostrados com fundo branco foram selecionados e salvos pelo usuário, portanto não irão mudar em um nova geração de FTs.

Na geração dos *thumbnails* iniciais, além dos parâmetros *mode* e opacidade máxima, uma outra variável influencia as imagens resultantes, e pode ser arbitrada pelo usuário na

ferramenta. Esse parâmetro é denominado *max materials* (número máximo de materiais) e informa quantos pontos de controle (dentre os 8 de cada vetor FT) irá receber um valor de opacidade diferente de zero. Por exemplo, se o valor de *max materials* é 5, apenas as 5 primeiras posições dos vetores FT irão receber valores de opacidade diferentes de zero. Isso é feito de modo a aumentar a diversidade dos resultados, além de favorecer valores *v* que possuam escores elevados. Além disso, em muitos volumes de dados a opacidade poderia se acumular demasiadamente dependendo da direção de visualização, ocultando assim estruturas importantes do volume. Um dos principais problemas da especificação de FTs é a sua dependência da aplicação e também do volume de dados que está sendo visualizado. Algumas combinações dos parâmetros mencionados anteriormente produziram melhores resultados quando aplicados a alguns volumes, mas não para todos os que foram testados.

Outro fator que afeta as imagens resultantes é a escolha de interpolar (ou não) a FT, já que essa interpolação pode atribuir opacidade a regiões que não as de interesse. No entanto, se apenas os voxels dos pontos de controle dos vetores FT possuírem uma opacidade diferente de zero na *lookup table* resultante, os resultados se assemelhariam aos obtidos através do *rendering* de isosuperfícies. Essa abordagem não faz parte do escopo desta dissertação, portanto, na ferramenta desenvolvida, a *lookup table* é sempre resultante de uma interpolação dos vetores FT.

Após a execução da dispersão inicial baseada nas bordas, e conseqüente exibição dos *thumbnails*, o usuário entra no primeiro nível de interação. É possível, então, aproximar, distanciar e rotacionar todos os *thumbnails* 3D interativamente, de modo a visualizar as imagens de diferentes pontos de vista. Caso as FTs iniciais não sejam satisfatórias para o usuário, o mesmo tem duas alternativas para refiná-las ainda no primeiro nível de interação.

Na primeira delas, é possível mudar os parâmetros de geração (*max opacity, max materials* e *thickness*) e gerar novamente todo o arranjo (ou um subconjunto) de *thumbnails*, executando a dispersão baseada nas bordas com os parâmetros modificados. O usuário pode salvar imagens selecionadas, que passam a ser exibidas em fundo preto (como mostra a Figura 4.6), sendo que é possível desfazer essa operação posteriormente. Os *thumbnails* salvos permanecem inalterados mesmo com a geração de uma nova população de FTs com diferentes parâmetros.

O segundo refinamento feito nesse nível de interação é através da seleção de um *thumbnail* de interesse e execução da dispersão baseada nas bordas com base nessa imagem. Dessa forma, o algoritmo decide randomicamente quantos e quais vetores FT sofrerão modificações para que se assemelhem ao vetor selecionado. Para isso, os valores *v* do vetor FT que gerou o *thumbnail* selecionado são copiados para os vetores a serem refinados. O restante dos parâmetros de geração (*max opacity, max materials* e *thickness*) são alterados, e essa alteração produz *thumbnails* ligeiramente semelhantes ao *thumbnail* selecionado. Cabe salientar que a alteração do parâmetro *thickness* afeta somente os vetores FT gerados através da tabela K, não importando o tipo de refinamento, pois apenas o método K depende desse parâmetro. Após terem sido feitos os refinamentos (opcionais) no primeiro nível de interação, o usuário pode selecionar um *thumbnail* de interesse e seguir para o segundo nível de interação da ferramenta, discutido na próxima seção.

4.5 Segundo Nível de Interação

Uma vez que o usuário selecionou uma imagem de interesse no primeiro nível de interação, a FT de cor e opacidade correspondente é usada para o *rendering* direto do volume, e o usuário pode então entrar no segundo (e opcional) nível de interação, ilustrado na Figura

4.7. É possível refinar a FT obtida e interagir com o volume em tempo real, rotacionando e aproximando ou afastando a imagem, do mesmo modo em que é feito com os *thumbnails*. A FT de opacidade é, então, exibido na mesma janela, juntamente com os pontos de controle do respectivo vetor FT. O eixo horizontal do gráfico representa os valores *v*, e o eixo vertical representa a opacidade. Nesse momento, o usuário pode escolher entre voltar para o primeiro nível de interação (os vetores FT restantes se mantêm inalterados) ou modificar o conjunto de parâmetros de geração e executar a dispersão baseada nas bordas novamente, com o resultado imediato na imagem volumétrica. Nesse nível é possível escolher qual o modo (*mode*) de preenchimento do vetor FT sendo visualizado (G, H ou K), visto que no primeiro nível essa escolha é automática, como discutido na seção 4.4.2.

Nesse nível de interação a cor, antes randomicamente atribuída aos pontos de controle dos vetores FT, pode ser escolhida pelo usuário. Acima do gráfico da função da FT na janela de *rendering*, é exibido um controle para escolha de cores básicas, denominado *color picker* (ilustrado nas figuras 4.7 e 4.8).

A decisão de implementar um *color picker* para a escolha de cores se baseou em interfaces comerciais conhecidas de edição de imagens, o que torna mais intuitivo para o usuário a identificação com a ferramenta. Esse controle funciona da seguinte forma: o usuário especifica o ponto do vetor FT que deseja mudar a cor, e escolhe uma nova no *color picker*. A cor escolhida é lida então do próprio *frame buffer* e copiada para o ponto de controle da FT, sobreescrevendo a existente. A *lookup table* é, então, interpolada novamente e atualizada, e o volume é exibido em tempo real com a nova FT.

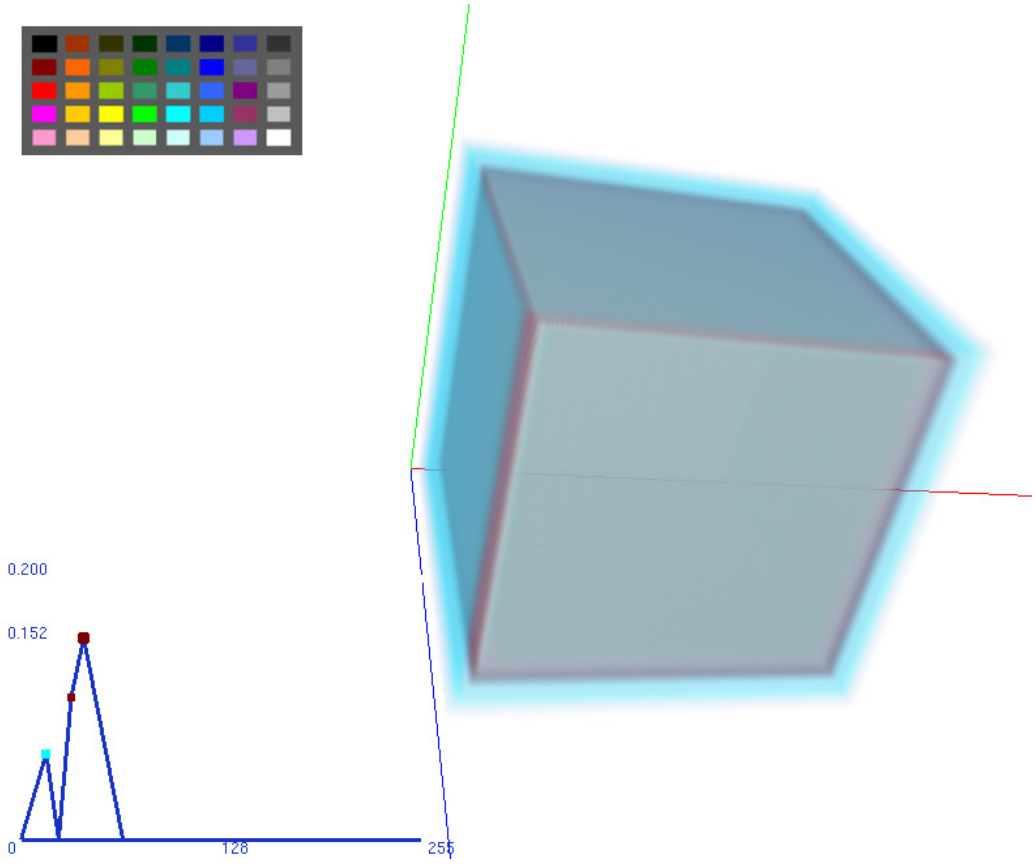
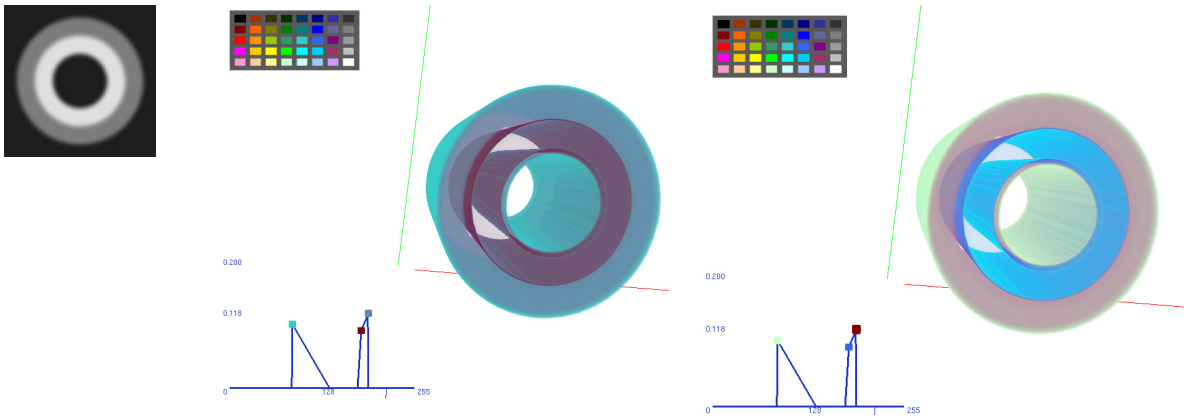


Figura 4.7: Interface do segundo nível de interação exibindo um volume sintético de 64^3 voxels. O gráfico e os pontos de controle da FT são exibidos na mesma janela.

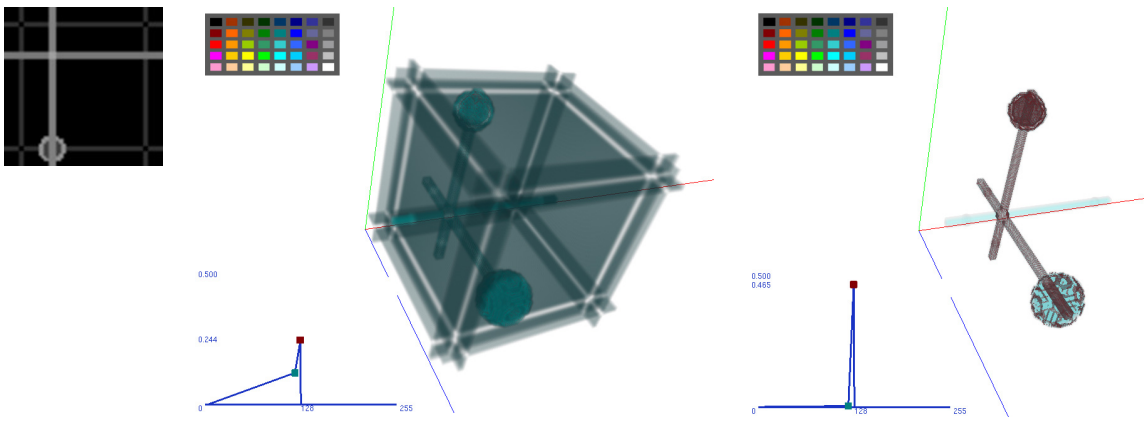
A Figura 4.8 ilustra o funcionamento da edição de cores. A Figura 4.8a representa uma fatia do volume de dados. A Figura 4.8b apresenta a imagem resultante utilizando a FT associada a um *thumbnail* selecionado no primeiro nível de interação. A imagem da Figura 4.8c foi obtida utilizando a mesma FT de opacidade (o gráfico permanece inalterado), mas com cores diferentes escolhidas no *color picker* após a seleção do ponto de controle.

Além da possibilidade de refinamento e alteração das cores no segundo nível de interação da ferramenta, é dada ao usuário a escolha de um refinamento mais "fino" na FT escolhida. Da mesma forma que na edição de cores, o usuário também pode aumentar ou diminuir a opacidade de cada ponto de controle dos vetores FT e obter novos resultados em tempo real. Apesar desse refinamento permitir ao usário uma edição praticamente manual da FT, é importante que o usuário tenha diversas formas de controle sobre o processo. A ferramenta proposta é semi-automática, garantindo que o usuário dispõe dessa alternativa, se a parte automática do processo não resulta na função mais adequada para as necessidades do mesmo. No entanto, é possível alterar a opacidade somente dos pontos de controle dos vetores FT. Ou seja, se um valor v não possuir um escore considerado elevado pelo algoritmo, ele não fará parte dos vetores, e portanto, sua opacidade será nula.



(a) (b) (c)

Figura 4.8: Interface do segundo nível de interação exibindo um volume de dados sintético de 64^3 voxels. (a) Fatia de amostra; (b) *Rendering* do volume (os pontos de controle da FT são exibidos com as cores atribuídas no primeiro nível de interação); (c) Volume exibido após mudança nas cores dos pontos de controle, mantendo as mesmas opacidades.



(a) (b) (c)

Figura 4.9: Interface do segundo nível de interação exibindo um volume de dados sintético de 64^3 voxels. (a) Fatia de amostra; (b) *Rendering* do volume (os pontos de controle da FT são exibidos com as cores atribuídas no primeiro nível de interação); (c) Volume exibido após mudança na FT (foi reduzida a opacidade em um dos pontos de controle e, no outro, aumentada).

4.6 Rendering

Para a implementação da visualização em tempo real do sistema proposto é utilizado o *rendering* volumétrico baseado em texturas (CABRAL et al., 1994; GELDER et al., 1996). O processo geral de *rendering* da ferramenta pode ser visto na Figura 4.10. Os dados volumétricos são armazenados em texturas 3D e renderizados através da exibição de uma geometria de amostragem, que consiste em planos ortogonais à direção de visualização (ENGEL, et al., 2002). Para a construção da geometria de amostragem, foi utilizado o algoritmo denominado *Weighted Sweep Graph* (WSG) (DIETRICH et al., 2004), que especifica os polígonos de acordo com a direção de visualização e interpola dinamicamente as coordenadas da textura 3D. Na abordagem proposta, o interesse estava em aplicações que pudessem manter uma taxa interativa de quadros por segundo. Por esse motivo, foi escolhido implementer um esquema de pós-classificação (WITTENBRINK et al., 1998; ENGEL et al., 2001; KRÜGER et al., 2003) utilizando recursos do hardware gráfico programável. A classificação é implementada como um programa de fragmentos executado na GPU. A geometria de amostragem construída pelo WSG (DIETRICH et al., 2004) atravessa o estágio de rasterização do *pipeline* gráfico. Nesse estágio, os polígonos resultam em fragmentos e as coordenadas de textura são interpoladas para cada fragmento gerado.

Ao invés dessas coordenadas serem usadas para buscar o valor de cor do fragmento em uma textura, as mesmas são usadas como índices para o acesso à *lookup table* onde está armazenada a FT (representada como uma textura 2D na memória da placa gráfica). Essa técnica é possível com o uso de um recurso demominado textura dependente, introduzido na terceira geração de GPUs. O fragmento recebe então os valores de cor e opacidade da FT e passa para o próximo estágio do pipeline, que compreende as operações por fragmento (ou operações *raster*). Nesse estágio, a operação mais importante para o *rendering* volumétrico é a composição (*blending*). Nessa operação, os fragmentos a serem escritos no *frame buffer* somam suas contribuições de cor e opacidade aos que já se encontram na imagem. O acúmulo de contribuições (ao invés de simplesmente sobrescrever o que já existia) permite a visualização de estruturas semitransparentes em um volume de dados. Finalmente, a imagem resultante é gerada, e permanecerá a mesma enquanto a FT ou o vetor de visualização permanecerem inalterados pelo usuário.

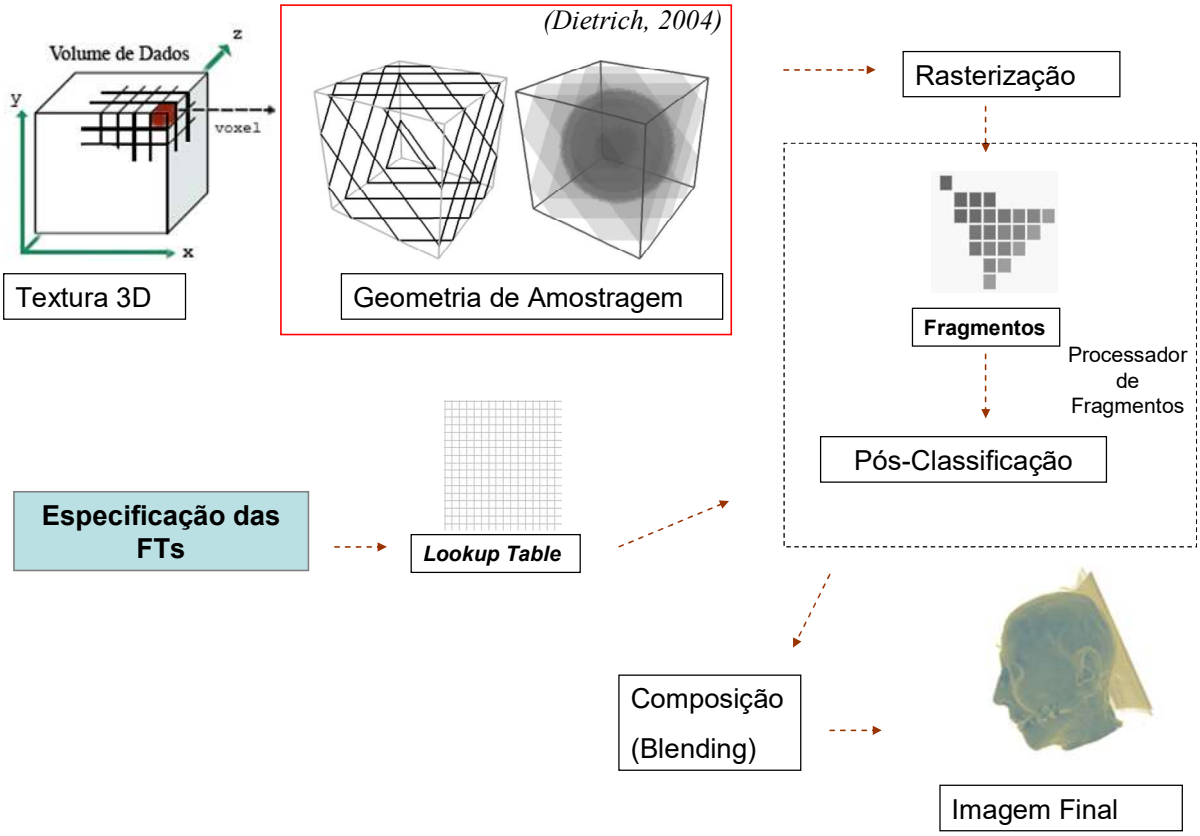


Figura 4.10: *Pipeline* do *rendering* volumétrico utilizado na ferramenta desenvolvida. O volume é armazenado em uma textura 3D, sendo então amostrado, classificado e composto no *frame buffer*, gerando a imagem final.

5 RESULTADOS

A ferramenta foi desenvolvida em linguagem C/C++, utilizando a biblioteca GLUI (GLUI, 2005), para a implementação da interface que provê controles como botões e campos editáveis (Figura 5.1). A programação do hardware gráfico foi feita usando Cg (FERNANDO et al., 2003).

Para testar a aplicação da ferramenta proposta, é feito um estudo de caso com volumes de dados sintéticos e reais, adquiridos através de TC e RM. Os resultados apresentados foram obtidos em um computador com um processador de 1GHZ, 512MB de memória e uma placa gráfica Nvidia® GeForce FX 5200 com 128MB de memória para o armazenamento das texturas. As dimensões dos volumes de dados utilizados variaram de 64^3 a 256^3 voxels. A interface permite ao usuário escolher entre quantidades pré-definidas de planos de amostragem, desde a configuração mais rápida, porém com imagens de qualidade baixa (40 planos) até a configuração mais lenta (1000 planos) porém com imagens de maior qualidade. As imagens mostradas foram exibidas com taxas interativas de cerca de 20 quadros por segundo.

A Figura 5.1 apresenta um volume de dados sintético que contém três materiais diferentes (uma fatia de amostra foi exibida na Figura 4.8a). Aqui é demonstrado que a ferramenta é capaz de distinguir os materiais, mostrando suas bordas, ou ainda isolar um ou mais materiais. Nesse exemplo não foi feito refinamento. Apesar da diversidade dos resultados, pode-se notar que não foi atribuída opacidade ao material mais externo, o que está de acordo com a proposta de realçar as regiões de transição do volume.

Basicamente, dois fatores afetam o desempenho da ferramenta: a resolução do volume de dados e o número de planos de amostragem utilizados no rendering volumétrico. Os volumes são armazenados como texturas na memória da placa gráfica, e as operações de transferência de dados da memória da CPU para a da GPU são relativamente custosas computacionalmente. Com relação aos planos de amostragem, quanto mais polígonos, mais fragmentos necessitam ser gerados na rasterização, prejudicando o desempenho do *rendering.* No entanto, volumes com altas resoluções e muitos planos de amostragem permitem o *rendering* de imagens de maior qualidade.

A Figura 5.2 apresenta o mesmo cenário, sendo que foram alterados os parâmetros de opacidade máxima, número de materiais e *thickness*, a fim de mostrar a diferença na geração das FTs iniciais.

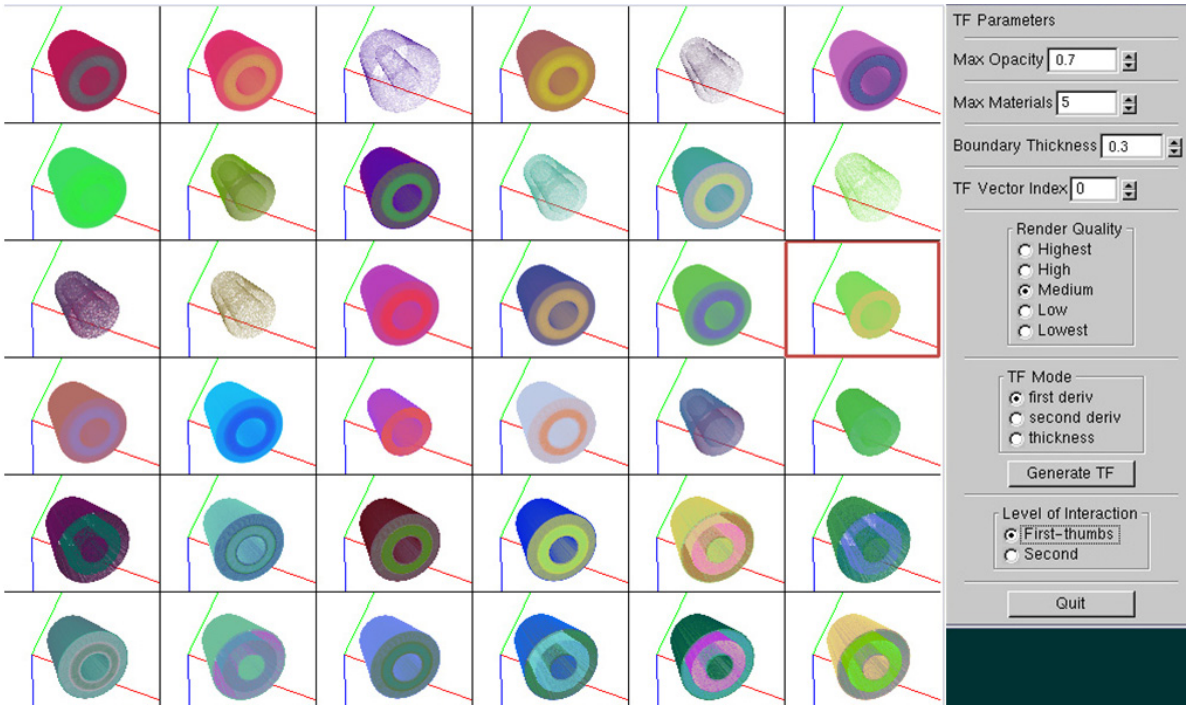


Figura 5.1: Primeiro nível de interação: volume de dados sintético (64^3 voxels). Janela de especificação de parâmetros: opacidade máxima de 0.7, máximo de 5 pontos de controle para cada vetor FT e *thickness* de 0.3.

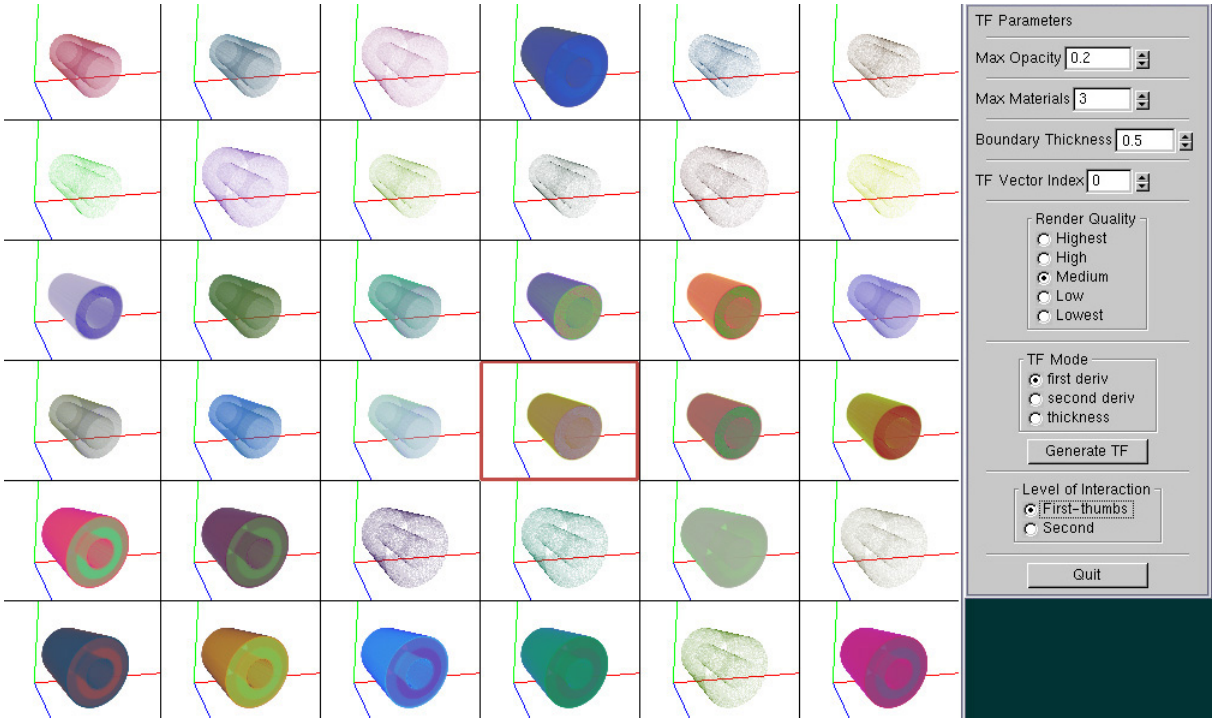
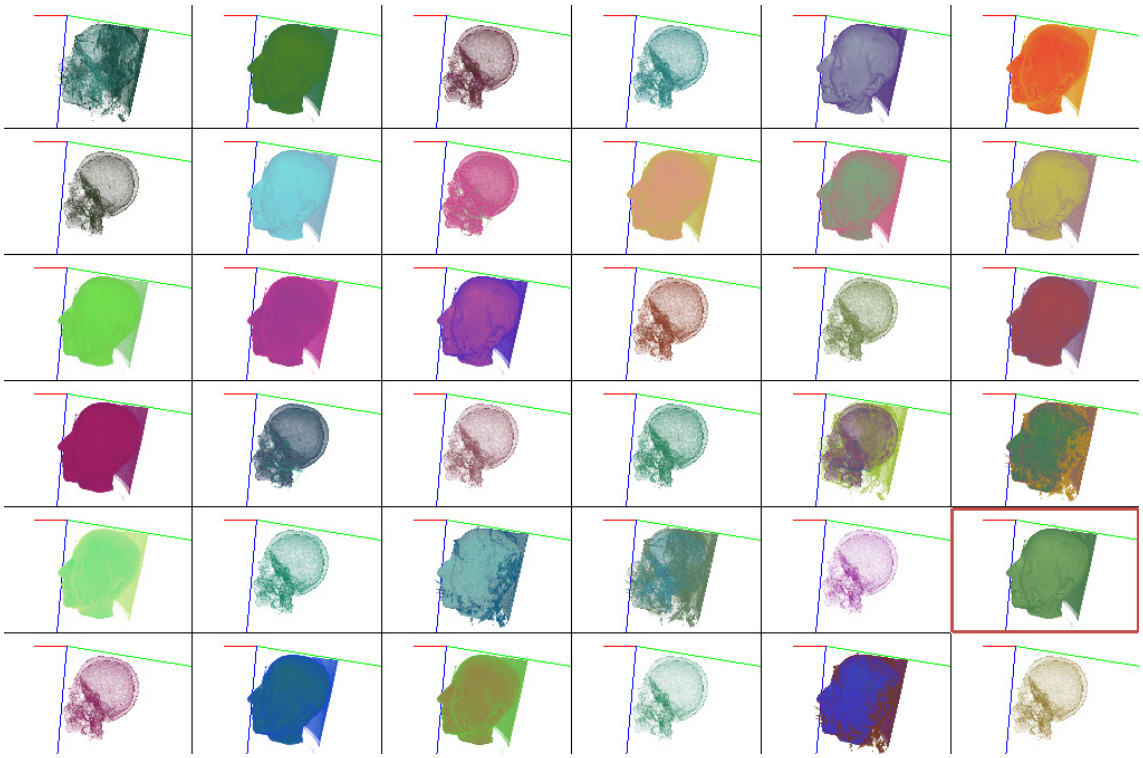


Figura 5.2: Primeiro nível de interação com o mesmo volume de dados da fig. 5.1. Parâmetros modificados para: opacidade máxima de 0.2, máximo de 3 pontos de controle para cada vetor FT e *thickness* de 0.5. Não foi feito refinamento nos *thumbnails*.

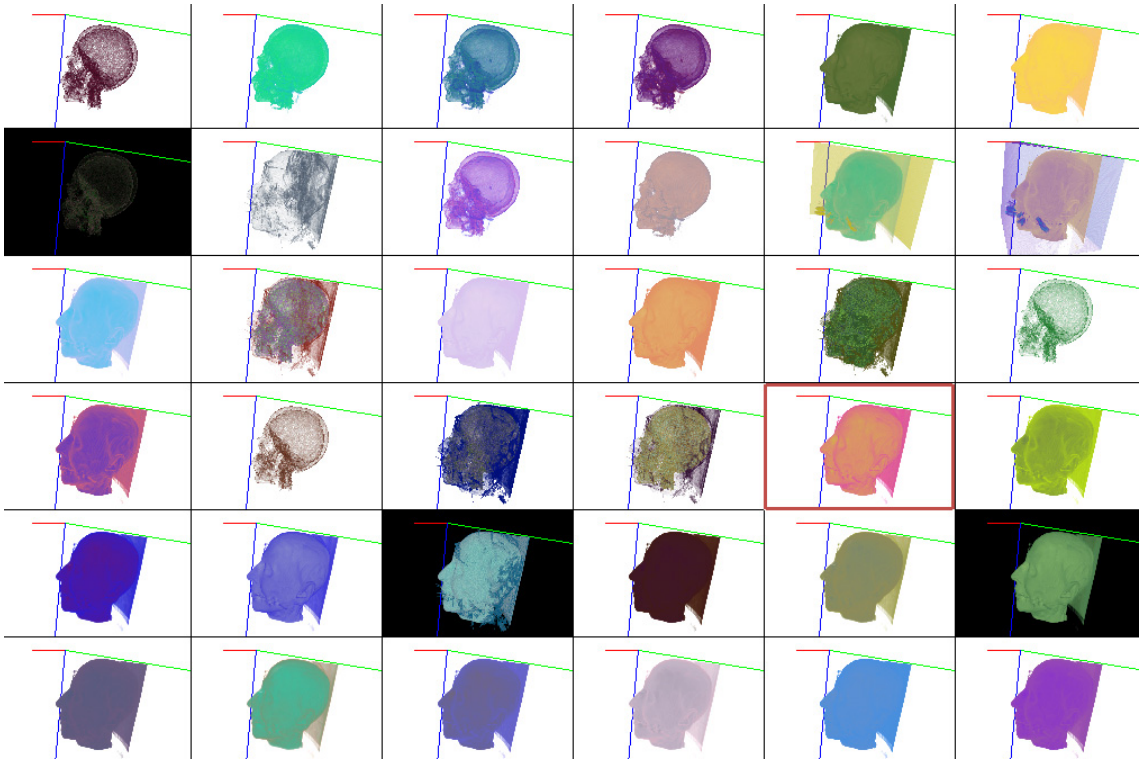
Pode ser observado que, com a opacidade máxima reduzida, os *thumbnails* se mostraram mais transparentes. Esse aspecto também foi resultado da redução dos pontos de controle, pois mais valores *v* que obtiveram bons escores receberam opacidade nula. O parâmetro *thickness* afeta somente os vetores gerados com o método K, dispostos nas duas últimas linhas da grade de *thumbnails*. Basicamente, quanto maior a *thickness,* mais largas ficam as bordas.

Na Figura 5.3 é demonstrada a ferramenta exibindo um volume de dados de TC conhecido (UNC Chapel Hill CT Head), com dimensões de 128^3. Nesta figura, o objetivo é mostrar o refinamento dos *thumbnails*. Inicialmente, a grade é construída com os parâmetros iniciais. Através de interações do usuário, alguns *thumbnails* de interesse são salvos. Então, os parâmetros são modificados e a dispersão baseada nas bordas é executada novamente, produzindo diferentes imagens. Novamente algumas imagens são salvas e a dispersão é executada, desta vez com base no *thumbnail* selecionado. A grade resultante apresenta características deste, no entanto incorporando as mudanças de parâmetros de geração.

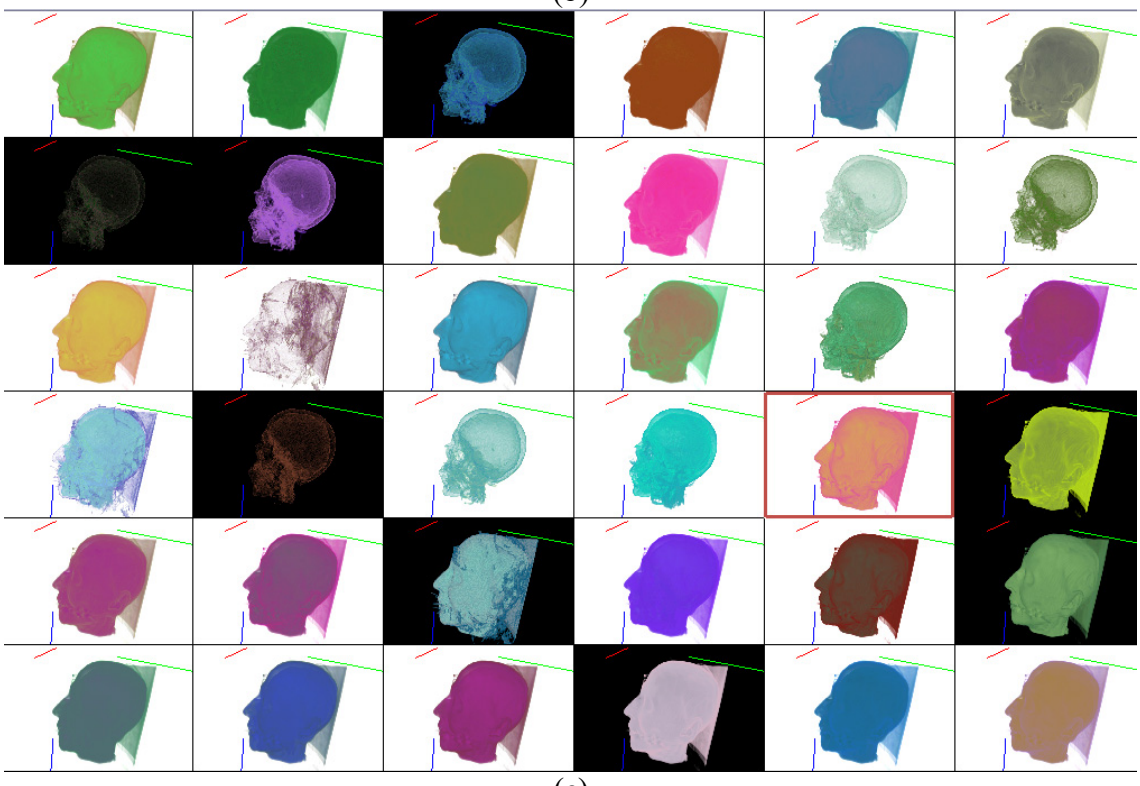
O conjunto de dados da Figura 5.3 apresenta desafios para as técnicas de especificação de FTs, visto que se trata de um volume de TC, contendo ruídos decorrentes da aquisição, além da baixa resolução, que produz suavizações forçadas pelo *rendering*. No entanto, pode-se observar que a ferramenta distingüe duas estruturas que possuem diferentes faixas de valores de voxel: a pele e os ossos. Na Figura 5.4 é exibido o mesmo volume de dados sendo utilizado no segundo nível de interação. É ilustrada a execução da dispersão baseada nas bordas após a mudança nos parâmetros de geração. Posteriormente, o usuário muda as cores dos pontos de controle e a última imagem é o resultado da alteração de opacidade nesses pontos.



(a)



(b)



(c)

Figura 5.3: Primeiro nível de interação exibindo um volume de dados de CT contendo 128^3 voxels. (a) *Thumbnails* iniciais sem refinamento; (b) alteração de parâmetros e geração de nova população de *thubmnails* (foram salvas algumas imagens, exibidas em fundo preto); (c) dispersão baseada nas bordas com base no *thumbnail* selecionado (borda da *viewport* destacada na cor vermelha).

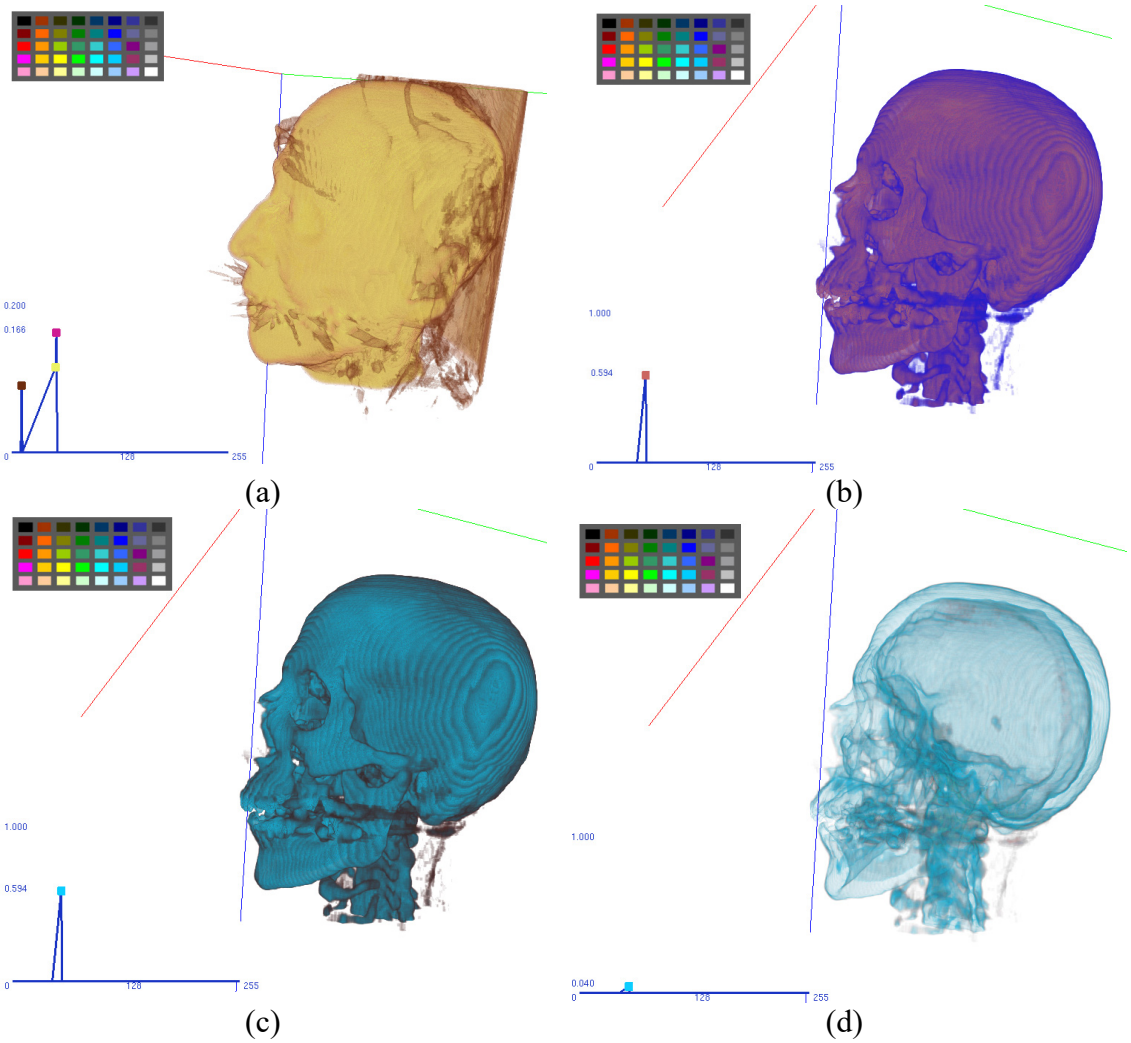


Figura 5.4: Segundo nível de interação exibindo um volume de dados de CT contendo 128^3 voxels. (a) Imagem resultante de um *thumbnail* selecionado no primeiro nível de interação; (b) alteração de parâmetros e geração de nova FT; (c) mudança da cor do ponto de controle; (d) redução da opacidade do ponto de controle.

A Figura 5.4a é resultado da seleção de um *thumbnail* do primeiro nível de interação, com os seguintes parâmetros: opacidade máxima de 0.7 e 3 diferentes pontos de controle do vetor FT. O usuário então modifica os parâmetros (opacidade máxima de 1 e 1 ponto de controle) e executa a dispersão baseada nas bordas utilizando o modo H (Figura 5.4b). A Figura 5.4c é resultado da mudança de cor através do *color picker.* Já na Figura 5.4d a opacidade no ponto de controle selecionado foi diminuída, o que tornou a imagem mais transparente, e permitiu a visão parcial do tecido cerebral. Cabe lembrar que essas regiões compartilham intervalos semelhantes de valores de dados, o que dificulta o realçe de uma região em particular pela FT.

A Figura 5.5 ilustra a utilização da ferramenta em outro volume de dados real, adquirido de uma RM de um joelho, também contendo 128^3 voxels. No primeiro nível de interação foram feitas aproximadamente cinco iterações do usuário (mudança de parâmetros e dispersões). Posteriormente, um *thumbnail* de interesse foi utilizado no segundo nível de

interação, tendo passado então por um refinamento nesse nível. Basicamente, imagens de RM apresentam desafios maiores que de CT, devido à sua característica mais ruidosa.

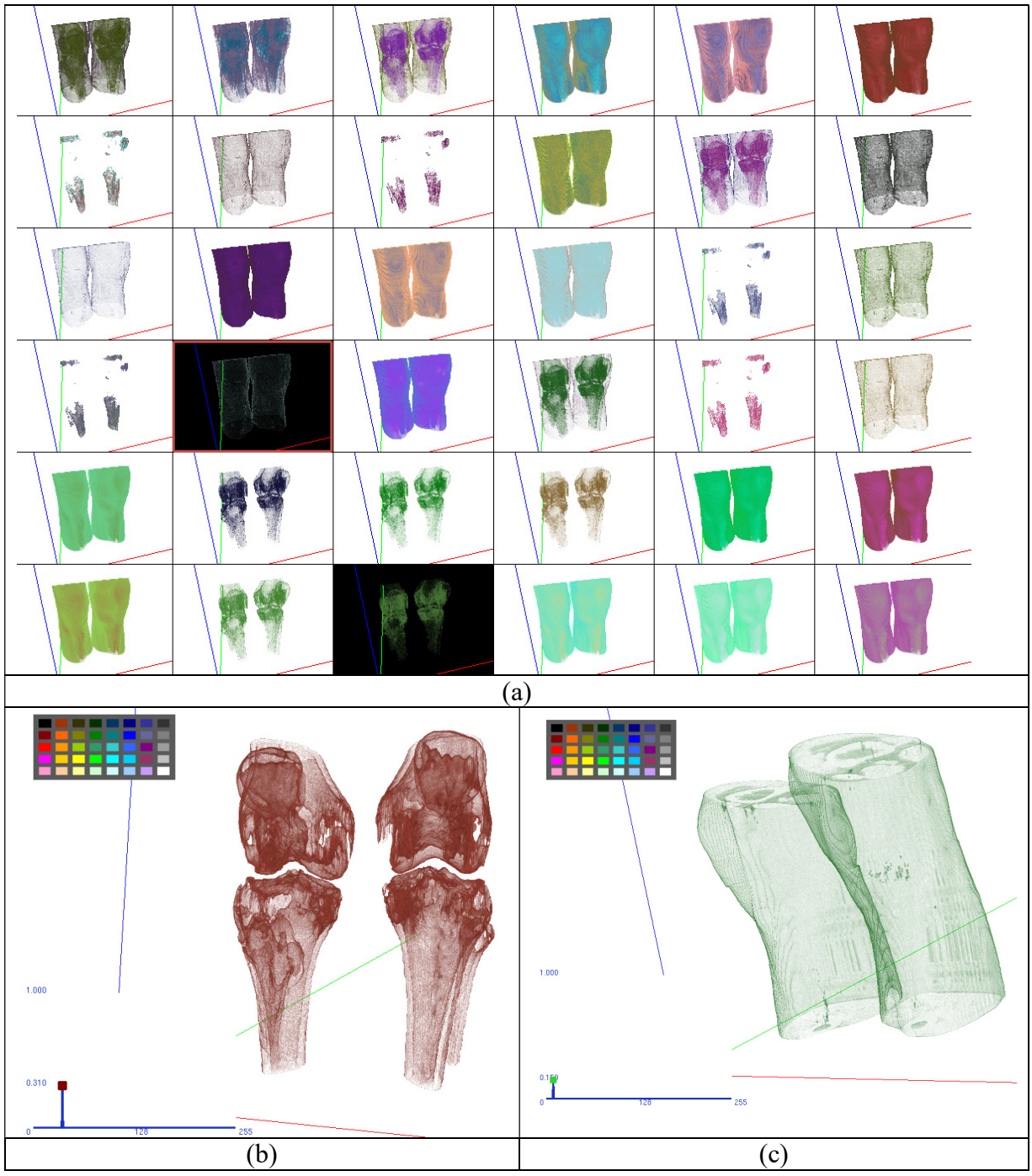


Figura 5.5: *Rendering* de um volume de dados de RM de um joelho contendo 128^3 voxels. (a) Grade de *thumbnails* com duas imagens salvas; (b) segundo nível de interação; (c) resultado da mudança de parâmetros e execução da dispersão baseada nas bordas.

Na Figura 5.6 é exibido outro volume de dados conhecido, o *engine block,* adquirido através de CT e com resolução de 256^3 voxels. A sequência de interações é semelhante à da Figura 5.5.

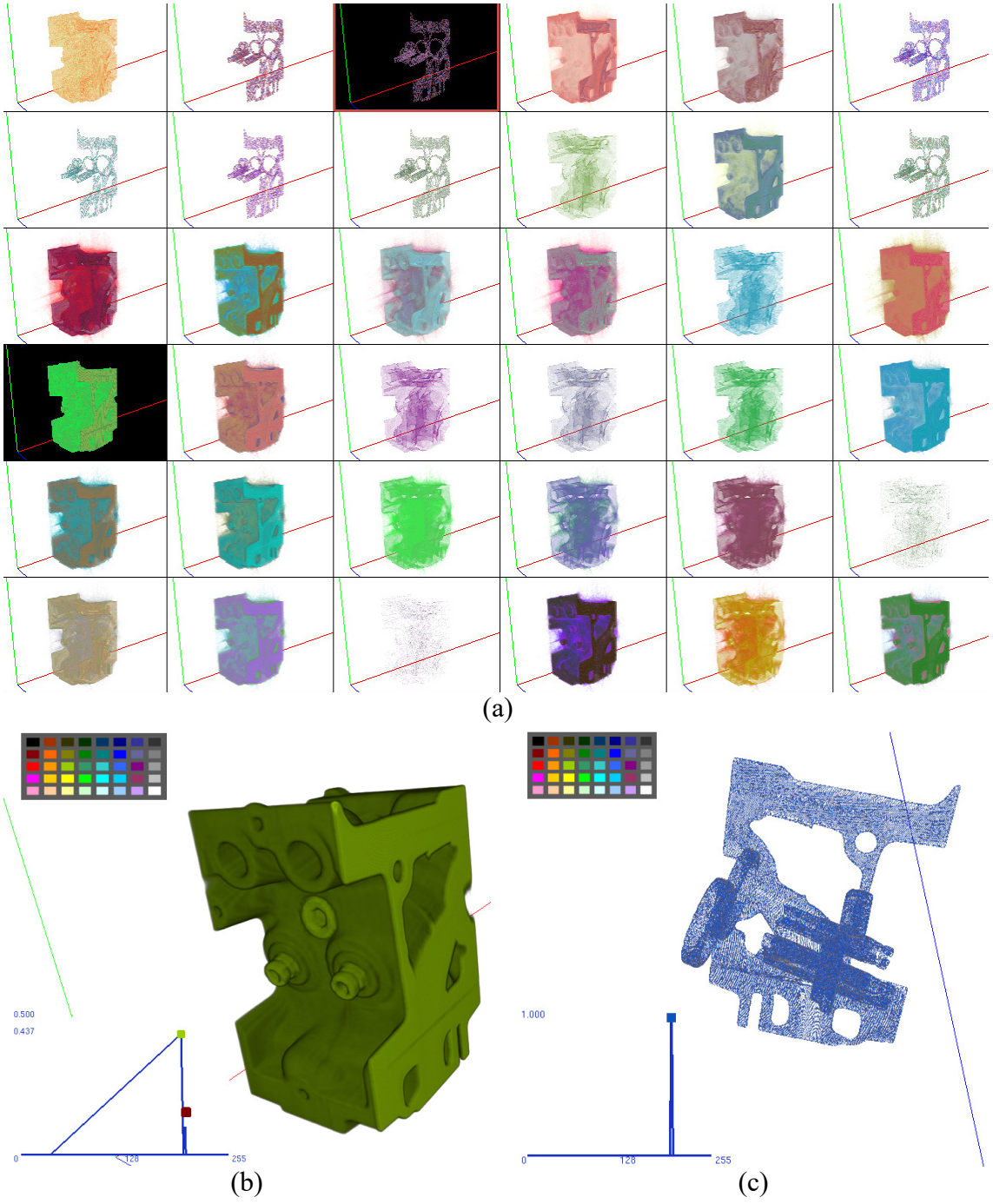


Figura 5.6: *Rendering* de um volume de dados de CT (*engine block*) contendo 256^3 voxels. (a) Grade de *thumbnails* com duas imagens salvas; (b) segundo nível de interação; (c) resultado da mudança de parâmetros e execução da dispersão baseada nas bordas.

Nas Figuras 5.6b e 5.6c é possível notar que o método proposto realçou diferentes materiais do volume no segundo nível de interação (a parte externa e interna da peça). A

opacidade não foi alterada após a dispersão baseada nas bordas. Alguns *thubmnails* apresentam artefatos ao redor da peça, em decorrência do grande número de materiais e regiões presentes nesse volume de dados.

Na Figura 5.7 são exibidos outros exemplos de volumes de dados utilizados com a ferramenta. Apenas o segundo nível de interação é mostrado, enfatizando os resultados finais obtidos. O intuito de exibir as bordas entre os diferentes materiais do volume é ilustrado nessa sequência de resultados. Novamente, não foram feitas alterações na opacidade dos pontos de controle que não através da dispersão baseada nas bordas.

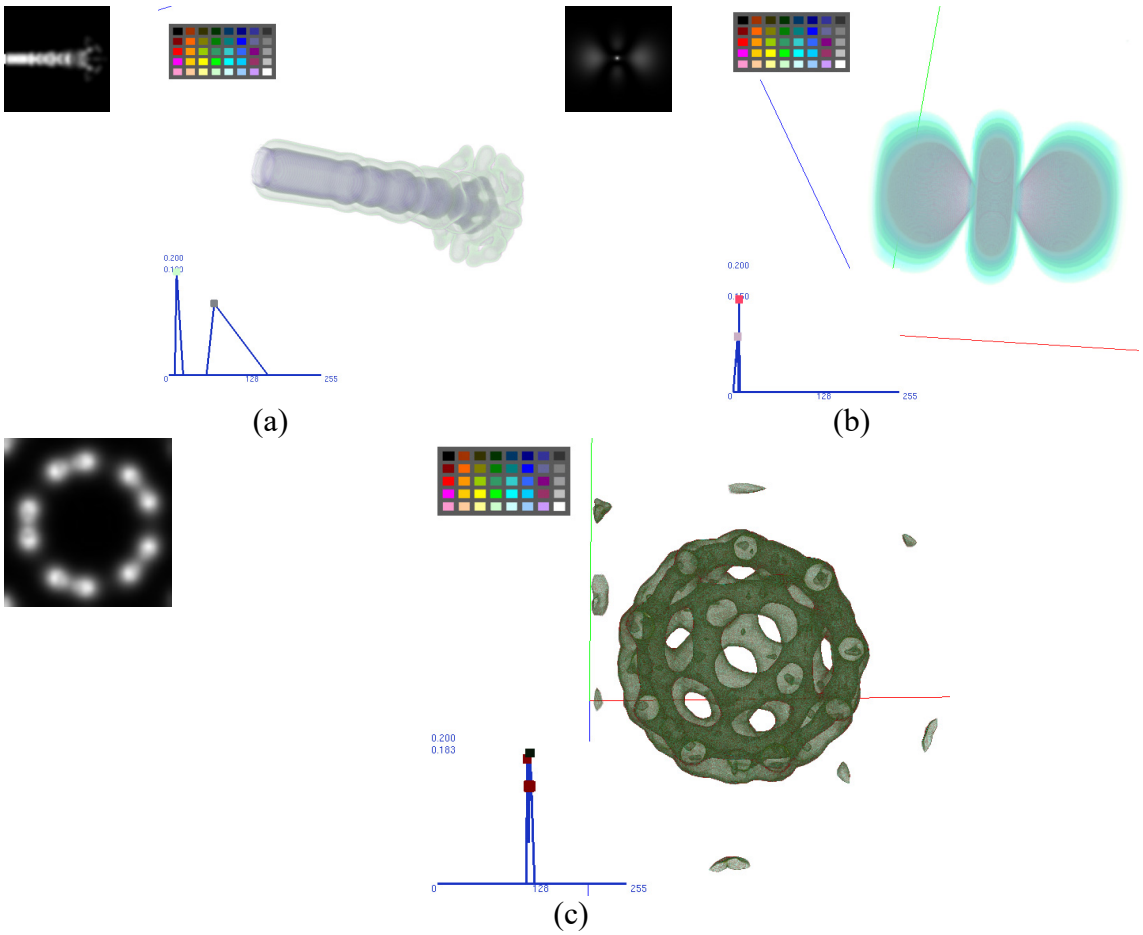


Figura 5.7: *Rendering* de um volume de dados contendo 64^3 voxels. (a) Fatia de amostra e *rendering* de um volume de simulação da injeção de combustível em um motor automobilístico; (b) fatia de amostra e *rendering* de um volume de simulação de um elétron de hidrogênio; (c) fatia de amostra e *rendering* de um volume de dados sintético.

6 DISCUSSÃO E CONCLUSÕES

6.1 Síntese e Contribuições

Nesta dissertação foi proposta uma ferramenta para especificação de funções de transferência para visualização volumétrica direta. Esse assunto, apesar de bastante explorado na literatura, encontra dificuldades tais como a dependência do volume de dados sendo visualizado. Inicialmente foram vistas técnicas simples, como "tentativa e erro" através da alteração de gráficos lineares, do tipo rampa ou exponenciais. Então, foram discutidos algoritmos que tentam auxiliar o usuário na definição da função de transferência, no intuito de tornar o processo mais intuitivo. Os recursos do *hardware* gráfico atual possibilitaram diversos avanços no rendering volumétrico em tempo real, e são utilizados na implementação da ferramenta proposta.

Comparado a outros métodos, especificamente Kindlmann e Durkin (KINDLMANN et al., 1998; KNISS et al., 2001; MARKS et al., 1997), a abordagem proposta se diferencia pelo uso dos valores de dados para guiar a geração das FTs de opacidade iniciais, e a possibilidade de atribuir cor no segundo nível de interação, sem a obrigatoriedade de especificação (manual) de um mapeamento direto ou indireto em um gráfico. Comparado ao *Design Galleries* (MARKS et al., 1997), o método proposto gera poucos thumbnails 3D, enquanto que aquele produz vários 2D. Além disso, no *Design Galleries* não há interação em 3D com os *thumbnails*. Com relação ao trabalho de Kindlmann e Durkin (KINDLMANN et al., 1998), o método proposto aqui emprega uma FT de cor enquanto que o mencionado lida apenas com FTs de opacidade. Além disso naquele método o usuário necessita especificar uma função (a função de ênfase nas bordas), que resulta em uma FT de opacidade.

No presente trabalho foram apresentadas técnicas para a especificação de funções de transferência para visualização volumétrica. Esses algoritmos (que podem ser classificados em *data-driven* e *image-driven*), quando comparados entre si, dão margem ao questionmento de qual nível de automação é mais apropriado, e quais ferramentas beneficiam os diferentes tipos de volumes de dados estudados atualmente.

O método *Design Galleries* (MARKS et al., 1997), por exemplo, é muito eficaz em achar uma função sem a necessidade do usuário conhecer a fundo o espaço das funções de transferência. No entanto, não gera ou apresenta diretamente as estruturas interiores do volume. Já o método *Contour Spectrum* (BAJAJ et al., 1997) consegue resolver esse problema, mas sua interface é útil somente após o entendimento das diferentes métricas utilizadas. O algoritmo de Kindlmann (KINDLMANN et al., 1998) apresenta bons resultados, no entanto restringe o usuário no espaço das funções de transferência. As abordagens de Kniss

(KNISS et al., 2003) e Tzeng (TZENG et al., 2003) são inovadoras e utilizam os recursos do hardware para calcular funções de transferência multidimensionais.

Uma limitação da ferramenta proposta é que a mesma não incorpora a implementação da técnica de pré-integração proposta por Engel (ENGEL et al., 2001), para produzir imagens de melhor qualidade com menos planos de amostragem. Isso se deve ao fato da preocupação maior na especificação das FTs e não no *rendering* volumétrico em si.

6.2 Trabalhos Futuros

Um possivel tema para trabalhos futuros seria a capacidade da ferramenta de lidar com FTs multidimensionais, visto que as mesmas podem classificar mais adequadamente materiais diferentes que compartilham intervalos semelhantes de volumes de dados. No entanto, a memória de texturas ainda é um gargalo das placas gráficas atuais, e essas funções demandam uma grande quantidade de memória. Além disso, a especificação de FTs 1D já apresenta dificuldades ao usuário, e se torna mais complexa ainda quando são utilizadas FT multidimensionais.

Outro ponto para futura pesquisa seria a utilização de outras técnicas para a detecção de bordas em um volume. No entanto, a primeira e segunda derivadas se mostraram eficientes e não demandam tantos cálculos complexos quanto em outros métodos de detecção de bordas.

REFERÊNCIAS

BAJAJ, C. L.; PASCUCCI, V.; SCHIKORE, D. The Contour Spectrum. In: IEEE SYMPOSIOUM ON VOLUME VISUALIZATION, 1997. **Proceedings**… [S.l.:s.n.], 1997. p. 167-173.

BOTHA, C; POST, F. Interactive Previewing for Transfer Function Specification in Volume Rendering. In: IEEE TCVG SYMPOSIUM ON VISUALIZATION, 2002. **Proceedings**… [S.l.:s.n.].

BRODLIE, K.; WOOD, J. Recent Advances in Volume Visualization. **Computer Graphics Forum**, Amsterdam, v. 20, p. 125-148, 2001.

CABRAL, B.; CAM, N.; FORAN, N. Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware, In: SYMPOSIUM ON VOLUME VISUALIZATION, 1994. **Proceedings…** [S.l.:s.n.], 1994. p.91-98.

DIETRICH, C.; NEDEL, L; OLABARRIAGA, S.; COMBA, J.; ZANCHET, D.; SILVA, A.; MONTERO, E. Real-time interactive visualization and manipulation of volumetric data using GPU-based methods. In: SPIE MEDICAL IMAGING, 2004. **Proceedings…**[S.l.:s.n.], 2004. p.182-192.

ENGEL, K.; KRAUS, M.; ERTL, T. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In: ACM SIGGRAPH/EUROGRAPHICS WORKSHOP ON GRAPHICS HARDWARE, 2001. **Proceedings…**, New York: ACM, 2001. p. 9-16.

ENGEL, K.; ERTL, T. Interactive high-quality volume rendering with flexible consumer graphics hardware. **ACM European Association for Computer Graphics Eurographics**. State of the Art Report…, ACM, 2002.

FERNANDO, R.; KILGARD, M. **The Cg Tutorial**: The Definitive Guide to Programmable Real-Time Graphics. Boston: Addison-Wesley, 2003.

FUCHS, H.; KEDEM, Z.; USELTON, S. Optimal Surface Reconstruction from Planar Contours. **Communications of the ACM**, New York, v.20, p.693-702, 1977.

GELDER, A. V.; KIM, K. Direct volume rendering with shading via three-dimensional textures. In: IEEE/ACM VOLUME VISUALIZATION SYMPOSIUM, 1996. **Proceedings…** [S. l.: s. n.], 1996. p. 23-30.

GLUI. GLUI User Interface Library. Disponível em: <http://www.cs.unc.edu/~rademach/glui/> Acesso em: maio 2005.

GONZALES, R.; WOODS, R. **Processamento de Imagens Digitais**. São Paulo: Blücher, 2000. 509 p.

HARRIS, M.; LASTRA, A. Real-Time Cloud Rendering. **Computer Graphics Forum**, v. 20, p. 76-84, 2001.

HAUSER, H. et. al. Two-level volume rendering. **IEEE Transactions on Visualization and Computer Graphics**, v. 7, p. 242-252, 2001.

HLADUVKA, J.; KONIG, A.; GROLLER E. **Curvature-Based Transfer Functions for Direct Volume Rendering**. Vienna: Vienna University of Technology, 2000. Technical Report

HÖNIGMANN, D.; RUISZ, J.; HAIDER, C. Adaptive Design of a Global Opacity Transfer Function for Direct Volume Rendering of Ultrasound Data. In: IEEE CONFERENCE ON VISUALIZATION, 2003. **Proceedings…** [S. l.: s. n.], 2003.

HOPF, M.; ERTL, T. Hierarchical Splatting of Scattered Data. In: IEEE CONFERENCE ON VISUALIZATION, 2003. **Proceedings…** [S. l.: s. n.], 2003.

KAUFMAN, A. **Volume Visualization**. Los Alamitos, CA: IEEE Computer Society Press, 1991. 479 p.

KAUFMAN, A.; COHEN, D.; YAGEL, R. Volume Graphics. **Computer**, v. 26, n. 7, p. 51–64. July 1993.

KEPPEL, E. Approximating Complex Surfaces by Triangulation of Contour Lines. **IBM Journal of Research and Development**, New York, v.19, n.1, p.2-11, 1975.

KINDLMANN, G.; DURKIN, J. W. Semi-automatic generation of transfer functions for direct volume rendering. In: IEEE CONFERENCE ON VISUALIZATION, 1998. **Proceedings…** [S. l.: s. n.], 1998. p. 79-86.

KINDLMANN, G. Transfer Functions in Direct Volume Rendering: design, Interface, Interaction. In: ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, SIGGRAPH, 2002. **Course Notes**. [New York, ACM], 2002. (Course 50).

KINDLMANN G. et al. Curvature-Based Transfer Functions for Direct Volume Rendering: Methods and Applications. In: IEEE CONFERENCE ON VISUALIZATION, 2003. **Proceedings…** [S. l.: s. n.], 2003. p. 67-74.

KNISS J.; KINDLMANN, G.; HANSEN, C. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In: IEEE CONFERENCE ON VISUALIZATION, 2001. **Proceedings…** [S. l.: s. n.], 2001. p. 255-262.

KNISS J. et al. Gaussian Transfer Functions for Multi-field Volume Visualization. In: IEEE CONFERENCE ON VISUALIZATION, 2003. **Proceedings…** [S. l.: s. n.], 2003. p. 497-504.

KÖNIG, A.; GRÖLLER, E. Mastering Transfer Function Specification by Using VolumePro Technology. In: SPRING CONFERENCE ON COMPUTER GRAPHICS, 2001. **Proceedings…**[S. l.: s. n.] 2001. p. 279-286.

KRÜGER, J.; WESTERMANN, R. Acceleration Techniques for GPU-based Volume Rendering. In: IEEE CONFERENCE ON VISUALIZATION, 2003, **Proceedings…** [S. l.: s. n.], 2003.

LACROUTE, P.; LEVOY, M. Fast volume rendering using a shear-warp factorization of the viewing transformation. In: ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, SIGGRAPH, 1994. **Proceedings…** New York: ACM, 1994. p. 451-458.

LEVOY, M. Display of surfaces from volume data. **IEEE Computer Graphics and Applications**, [S. l.], v. 8, p. 29-37, May 1988.

LEVOY, M. Efficient ray-tracing of volume data. ACM Transactions on Graphics, [S. l.], v. 9, p. 245-261, July 1990.

LICHTENBELT, B.; CRANE, R.; NAQVI, S. **Introduction to Volume Rendering**. Upper Saddle River, NJ: Prentice Hall, 1998.

LORENSEN W. E.; CLINE H. E. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In: INTERNATIONAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, 1987. **Proceedings…** New York: ACM, 1987, p. 163–169.

MANSSOUR, I. H.; FURUIE, S. S.; OLABARRIAGA, S. D.; FREITAS, C. M. D. S. Visualizing Inner Structures in Multimodal Volume Data. In: SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO GRÁFICA E PROCESSAMENTO DE IMAGENS, SIBGRAPI, Fortaleza. **Anais...** [S.l: s.n], 2002

MARKS, J. et al. Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation. In: ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, SIGGRAPH, 1997. **Proceedings…** [S.l: s.n], 1997. p. 389-400.

MAX, N. Optical models for direct volume rendering. **IEEE Transactions on Visualization and Computer Graphics**, [S. l.], v. 1, June 1995.

MEISSNER, M.; HOFFMANN, U.; STRASSER, W. Enabling classification and shading for 3D texture mapping based volume rendering using OpenGL and extensions. In: IEEE CONFERENCE ON VISUALIZATION, 1999. **Proceedings…** [S.l: s.n], 1999. p. 207-214.

MUELLER, K.; MÖLLER, T.; CRAWFIS, R. Splatting Without the Blur. In: IEEE Conference on Visualization, 1999. **Proceedings…** [S.l: s.n], 1990. p. 363–370.

MUELLER K. et al. High-Quality Splatting on Rectilinear Grids with Efficient Culling of Occluded Voxels. IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, v. 1, [S.l.], p. 116-135, June 1999.

PFISTER H. et al. Generation of transfer functions with stochastic search techniques. In: IEEE CONFERENCE ON VISUALIZATION, 1996, **Proceedings…** [S.l: s.n], 1996. p. 227-234.

PFISTER, H. et al. The VolumePro Real-Time Ray-Casting System. ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, SIGGRAPH, 1999. **Proceedings…** [S.l: s.n], 1999. p. 251-260.

PFISTER H. et al. The Transfer Function Bake –off. IEEE COMPUTER GRAPHICS AND APPLICATIONS, v.21, p.16-22, June 2001.

PRAUCHNER, J.; DIETRICH, C.; NEDEL, L.; FREITAS, C. Modelagem, Animação e Rendering de Nuvens utilizando Visualização Volumétrica. In: SBC SYMPOSIUM ON VIRTUAL REALITY, 2004, São Paulo. **Proceedings…** São Paulo: SBC, 2004. p. 100-110.

ROETTGER S. et al. Smart Hardware-Accelerated Volume Rendering. In: IEEE CONFERENCE ON VISUALIZATION, 2003, **Proceedings…** [S.l: s.n], 2003. p. 231-238.

SILVA, I. Avaliação da Qualidade de Imagens Médicas Geradas por Ray Casting. Porto Alegre, PGCC da UFRGS, julho/2003. (Dissertação de mestrado).

SILVA, M.R.M. Alternativas de Visualização de Volumes Baseadas em Ray Casting. Porto Alegre, PGCC da UFRGS, março/2000. (Dissertação de mestrado).

SHREINER, D. **OpenGL reference manual**: the official reference document to OpenGL, version 1.2. [S.l.]: Addison-Wesley, 1999.

SRIVASTAVA, V.; CHEBROLU, U.; MUELLER, K. Interactive Transfer Function Modification for Volume Rendering Using Pre-Shaded Sample Runs. In: PACIFIC CONFERENCE ON COMPUTER GRAPHICS AND APPLICATIONS, 2002. **Proceedings…** [S.l: s.n], 2002. p. 489-490.

SWAN, E. et al. An Anti-Aliasing Technique for Splatting. In: IEEE CONFERENCE ON VISUALIZATION, 2003, **Proceedings…** [S.l: s.n], 2003. p. 197-204.

TZENG, F.; LUM, E.; MA, K. A Novel Interface for Higher-Dimensional Classification of Volume Data. In: IEEE CONFERENCE ON VISUALIZATION, 2003, **Proceedings…** [S.l: s.n], 2003.

UDUPA J.K.; ODHNER, D. Shell Rendering. IEEE COMPUTER GRAPHICS AND APPLICATIONS, [S.l.], v. 13, 1993.

UPSON C.; KEELER M. V-Buffer: Visible Volume Rendering. In: INTERNATIONAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, 1988. **Proceedings…** New York: ACM, 1988. p. 59–64.

WESTENBERG, M.; ROERDINK, J. X-Ray Volume Rendering by Hierarchical Wavelet Splatting. In: INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION, 15., 2000. **Proceedings…** [S.l: s.n], 2000. p. 163-166.

WESTERMANN, R.; SEVENICH, B. Accelerated Volume Ray-Casting Using Texture Mapping. In: IEEE CONFERENCE ON VISUALIZATION, 2001. **Proceedings…** [S.l: s.n], 2001. p. 271-278.

WESTOVER, L. Interactive Volume Rendering. In: WORKSHOP ON VOLUME VISUALIZATION, 1989. Proceedings. . . [S.l.], University of North Carolina Press, 1989. p. 9–16.

WESTOVER, L. Footprint Evaluation for Volume Rendering. In: ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, SIGGRAPH, 1990. **Proceedings…** [S.l: s.n], 1990. p. 367–376.

WILHELMS, J.; GELDER, A. Van. A Coherent Projection Approach for Direct Volume Rendering. In: ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, SIGGRAPH, 1991. **Proceedings…** [S.l: s.n], 1991. p. 275–284.

WITTENBRINK, C. M.; MALZBENDER, T.; GROSS, M. E. Opacity-weighted color interpolation for volume sampling. In: IEEE SYMPOSIOUM ON VOLUME VISUALIZATION, 1988. **Proceedings…** [S. l.: s. n.], 1988.

ZUIDERVELD, K. et al. Multimodality Visualization of Medical Volume Data. **Computer and Graphics**, Oxford, v.20, n.6, p.775-791, 1996.

APÊNDICE A *PIXEL SHADER* RESPONSÁVEL PELA EXECUÇÃO DO PROCESSO DE PÓS-CLASSIFICAÇÃO

```
struct vert2frag
{
      float3 texcoord0 : TEXCOORD0;
};

void main
(
      vert2frag               v2f
      , uniform sampler3D     textureA
      , uniform sampler2D     textureB
      , out float4 Col : COLOR
      )
{

      float4 colorA = tex3D(textureA, v2f.texcoord0);

      float4 colorB = tex2D(textureB, float2(colorA.r,0.5f));
      //0.5f serve para evitar interpolação com a borda da textura 2d


      Col = colorB;
}
```