

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito: 22.04.2003

Assinatura: *Ana Paula Jampão Jr/enc*

**Técnica híbrida de visualização para
exploração de dados volumétricos não
estruturados**

Patricia Shirley Herrera Cateriano

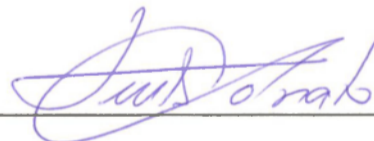
Orientador: Prof. Dr. Luis Gustavo Nonato

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências de Computação e Matemática Computacional.

USP – São Carlos
Abril/2003

A Comissão Julgadora:

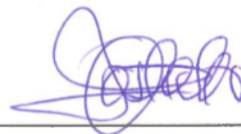
Prof. Dr. Luis Gustavo Nonato



Prof. Dr. João Luiz Dihl Comba



Prof. Dr. Antonio Castelo Filho



Dedico este trabalho aos meus queridos pais Julio Alberto Herrera Banda., Binda Cateriano de Herrera, e aos meus irmãos Julio Cesar e Yanina, pelo amor e apoio constante.

A meus tios Alberto, Rosa, José, Reina, a minha prima Giuliana, a toda minha família, as minhas melhores amigas e em especial a Deus, por ser sempre incondicionais e fontes de inspiração.

AGRADECIMENTOS

Agradeço a todos aqueles que, direta ou indiretamente, colaboraram para a realização deste trabalho. Em particular agradeço:

Ao Professor Dr. Luis Gustavo Nonato, pela sua valiosa orientação, apoio e confiança na realização da pesquisa.

A Professora Dra. Agma Traina, pelo apoio e ajuda generosa no início do meu mestrado.

A todos os colegas do LCAD pela colaboração proporcionada: Alex, Helton, Vinicius.

Aos professores e funcionários do Departamento de Estatística e Computação que contribuíram para a realização desse trabalho.

Ao programa de Pós-graduação da área de Ciências da Computação e Matemática Computacional do ICMC – USP, pela dedicação e orientação que dispõem a seus alunos.

A Jorge pelas valiosas sugestões durante a fase experimental.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico CNPq, pela concessão da bolsa de estudo de mestrado.

A todos os amigos e colegas que fiz na cidade de São Carlos.

A todos aqueles que contribuíram para a concretização deste trabalho, os meus sinceros agradecimentos.

SUMÁRIO

LISTA DE FIGURAS.....	i
RESUMO.....	iii
ABSTRACT.....	iv
1. Introdução.....	1
2. Visualização Volumétrica.....	6
2.1. Considerações Iniciais.....	6
2.2. Descrição do Volume.....	7
2.3. Classificação dos Algoritmos de Visualização Volumétrica.....	10
2.4. Algoritmos de Rendering Superficial.....	12
2.5. Algoritmos de Rendering Volumétrico Direto.....	18
2.6. Considerações Finais.....	26
3. Técnicas de Visualização Híbridas.....	27
3.1. Considerações Iniciais.....	27
3.2. Os Algoritmos de Rendering Híbrido.....	27
3.2.1. Hibridização Segundo o Tipo de Rendering.....	28
3.2.2. Hibridização Segundo o Domínio do Volume.....	32
3.2.3. Hibridização Segundo a Arquitetura de Computação.....	34
3.3. Otimizações.....	35
3.4. Considerações Finais.....	40
4. Visualização Volumétrica Baseada no Ray Casting.....	41
4.1. Considerações Iniciais.....	41
4.2. Trabalhos Correlatos.....	43
4.3. O Algoritmo de Ray Casting Binário.....	46
4.4. O Algoritmo de Ray Casting.....	47
4.4.1. Classificação.....	50
4.4.2. Iluminação.....	51

4.4.3. Projecção.....	52
4.5. Otimizações sobre o Algoritmo Original.....	53
4.5.1. Lançamento dos Raios.....	54
4.5.2. Determinação dos Segmentos dos Raios que Interceptam o Volume.....	56
4.5.3. Exploração da Coerência Espacial.....	58
4.5.4. Acumulação das Contribuições de Opacidade e Cor ao Longo do Raio.....	60
4.5.5. Exibição do Plano de Visualização Durante a “Construção” da Imagem.....	61
4.5.6. Aceleração Utilizando Algoritmos Paralelos.....	62
4.6. Considerações Finais.....	64
5. Uma Técnica Híbrida de Visualização para a Exploração de Dados Volumétricos Não Estruturados.....	65
5.1. Considerações Iniciais.....	65
5.2. Pipeline de Visualização Híbrido.....	66
5.2.1. Armazenamento de Dados Volumétricos.....	68
5.2.2. Ray Casting.....	69
5.2.3. Rendering de Superfícies.....	76
5.3. Implementação.....	77
5.4. Considerações Finais.....	82
6. Conclusões e Sugestões para Trabalhos Futuros.....	84
REFERERÊNCIAS BIBLIOGRÁFICAS.....	86

LISTA DE FIGURAS

Figura 1.1.	Representação dos grupos de tipos de dados	3
Figura 2.1.	Unidades de Representação Volumétrica.....	8
Figura 2.2.	Interpolação de ordem zero.....	9
Figura 2.3.	Interpolações de ordem superior.....	9
Figura 2.4.	Diferentes formas poliedrais adotadas como células.....	9
Figura 2.5.	<i>Pipeline</i> de Visualização Volumétrica.....	11
Figura 2.6.	<i>Rendering</i> Superficial - faces das células.....	14
Figura 2.7.	<i>Marching Cubes</i> : as configurações básicas para triangulação de uma única célula.....	15
Figura 2.8.	Algoritmo <i>Marching Tetraedra</i> : configuração dos vértices e dos cortes correspondentes às iso-superfícies.....	17
Figura 2.9.	<i>Rendering</i> Superficial por triangulação de contornos.....	18
Figura 2.10.	<i>Rendering</i> volumétrico utilizando o teorema de projeção de fatias de Fourier.....	21
Figura 2.11.	Algoritmo de <i>Ray Casting</i>	24
Figura 2.12.	Algoritmo <i>Shear-Warp</i>	25
Figura 3.1.	Estrutura de dados “ <i>Face Octree</i> ”.....	30
Figura 3.2.	<i>Shear-Warp</i> Híbrido.....	31
Figura 3.3.	Métodos híbridos baseados em <i>wavelets</i>	33
Figura 3.4.	O paralelismo do espaço do objeto e do espaço da imagem.....	35
Figura 3.5.	Pirâmide de Representação de Volume.....	36
Figura 3.6.	Refinamento Progressivo de Hugues Hopes.....	37
Figura 3.7.	<i>Rendering</i> volumétrico paralelo, por Anton Koning	39
Figura 4.1.	Projeção paralela e perspectiva.....	46
Figura 4.2.	Técnica bidimensional por etapas.....	47
Figura 4.3.	Algoritmo de <i>Ray Casting</i>	48
Figura 4.4.	<i>Pipeline</i> de rendering direto de Levoy.....	50
Figura 4.5.	<i>Ray Casting</i> discreto. Base do algoritmo “ <i>Template-based</i> ”	

	<i>volume viewing</i>	55
Figura 4.6.	Redução do número de pontos de amostra necessários, ao longo do raio.....	57
Figura 4.7.	Representação da árvore <i>BSP</i>	59
Figura 4.8.	Construção de uma <i>Octree</i>	60
Figura 4.9.	Construção da imagem por refinamento e progressão da imagem.....	62
Figura 5.1.	<i>Pipeline</i> de Visualização Híbrido.....	67
Figura 5.2.	Cone de luz.....	71
Figura 5.3.	Cálculo das interseções de um raio.....	72
Figura 5.4.	Casos de interseção do raio com uma face.....	73
Figura 5.5.	Cálculo de iluminação nas células.....	73
Figura 5.6.	Obtenção do valor do ponto amostrado mediante interpolação....	74
Figura 5.7.	Composição de cor e opacidade.....	75
Figura 5.8.	Determinação da cor da face.....	76
Figura 5.9.	Esquema da Estrutura de Dados.....	77
Figura 5.10.	Série de imagens de uma esfera dentro de um cubo.....	80
Figura 5.11.	Serie de imagens de três esferas, cada uma dentro da outra.....	82

RESUMO

HERRERA, P. S. (2003). Técnica Híbrida de Visualização Volumétrica para Exploração de Dados Volumétricos Não Estruturados. São Carlos, 2003. 92p. Dissertação (Mestrado) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo.

Este trabalho apresenta uma nova técnica de visualização que aproveita as vantagens do *rendering* volumétrico direto e do *rendering* de superfícies em um ambiente híbrido. O método faz uso de uma pré-visualização sobre o bordo do volume que viabiliza uma interação em tempo real com objetos volumétricos modelados por meio de malhas não estruturadas. Além disso, essa nova abordagem de visualização é paralelizável e pode se acelerada com placas gráficas comuns.

Palavras-chave: *rendering* volumétrico híbrido, malhas não estruturadas, *Ray Casting*.

ABSTRACT

HERRERA, P. S. (2003). *Hybrid Technique of Volume Rendering to the Exploration of Unstructured Volumetric Data*. São Carlos, 2003. 92p. Dissertação (Mestrado) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo.

This work presents a new visualization technique that exploits the advantages of direct volume rendering and surface rendering in a hybrid environment. The method developed here makes use of a pre-visualization on the volume boundary to enable real time interaction with unstructured volumetric meshes. Furthermore, this new visualization approach can be implemented on existing parallel architectures and speed up by conventional graphical hardware.

Keywords: hybrid volume rendering, unstructured volume data, Ray Casting.

CAPITULO 1

Introdução

O grande avanço tecnológico dos últimos anos propiciou o crescimento de aplicações que produzem dados volumétricos. Alguns exemplos são as tomografias computadorizadas, a microscopia e as medições sísmicas. A quantidade de informações geradas por estas aplicações já é imensa e cresce a cada dia. Todo esse volume de informação precisa ser analisado e interpretado de forma simples, clara e rápida. Para tornar mais ágil o processo científico de descoberta, confirmação e reprodução desses dados, utilizam-se como ferramentas os métodos de Visualização Científica.

Visualização Científica é o nome formal dado ao campo da ciência da computação que engloba linhas de pesquisas como: interface do usuário, representação dos dados e algoritmos de processamento, representações visuais, e outras apresentações sensoriais que ajudem aos usuários compreender seus dados [MDB87, BCEG91]. Muitas vezes, a visualização permite observar fenômenos que não poderiam ser estudados, ou que dificilmente seriam analisados sem o auxílio do computador e dos recursos visuais associados. A simulação de um fluido simples, por exemplo, pode gerar bilhões de números difíceis de serem interpretados sem o uso de técnicas gráficas.

Dentro da Visualização Científica, uma das áreas de maior pesquisa atualmente é a Visualização Volumétrica. A Visualização Volumétrica tem sido usada tanto para extrair informações significativas dos conjuntos de dados volumétricos, como para representar, manipular e “renderizar” tais dados [DCH88,Le90a,Ka91,Wi91].

Para serem úteis, as técnicas de Visualização Volumétrica devem oferecer representações compreensíveis, manipulação eficiente, e visualização rápida dos dados. Poucos sistemas atuais

oferecem todos esses recursos; por isso, o desenvolvimento e refinamento de algoritmos de visualização volumétrica tem atraído a atenção de muitos pesquisadores.

Tipicamente, os algoritmos de Visualização Volumétrica dividem-se em dois grupos: as técnicas de *Rendering* Superficial (RS) e as técnicas de *Rendering* Volumétrico Direto (RVD).

As técnicas de RS apresentam uma abordagem bastante conhecida de visualização volumétrica, a extração de iso-superfícies mediante interpolações dos elementos do volume. Elas utilizam aproximações poliedrais baseadas em primitivas geométricas (geralmente planas) tais como pontos, linhas e polígonos.

Já as técnicas de RVD tornaram-se populares por apresentar imagens realísticas e de alta qualidade. Estas técnicas de visualização não utilizam representações geométricas intermediárias, como os polígonos usados nas técnicas de RS, evitando problemas de inconsistência geométrica e topológica provocados por erros de aproximação. Além disso, como todo o conjunto de dados é usado, preserva-se a noção global das características dos dados.

Os métodos de RS são mais rápidos que os de RVD, já que percorrem o volume uma única vez para extrair as superfícies [E192].

Em muitas aplicações, é necessário não apenas visualizar os dados que definem o volume, mas também modelar estruturas internas ao volume geradas por eles. Por isso, torna-se bastante útil o desenvolvimento de algoritmos híbridos que permitem a visualização simultânea de dados volumétricos e estruturas modeladas mediante representações poligonais dos dados. Exemplos de aplicações que requerem a utilização de algoritmos híbridos de visualização volumétrica [RCH99] se encontram na medicina, como no desenvolvimento de implantes e próteses que requerem da utilização de modelos geométricos para serem devidamente encaixados no volume [Le90a] ou no planejamento dos tratamentos de radiação [L.FPR90], onde os tumores são volumes visualizados a partir de conjuntos de dados MRI, e os caminhos dos feixes de radiação (definidos como cilindros ou cones) são adicionados à cena para assegurar que o caminho correto do feixe até o tumor. Existem poucas técnicas que explorem a hibridização dos algoritmos de RS e RVD, sendo um dos nossos interesses desenvolver uma nova técnica deste tipo. As definições mais detalhadas das técnicas de RS, RVD e dos algoritmos híbridos, são apresentadas em capítulos subsequentes.

Como foi mencionado, as técnicas de RS e RVD utilizam varias formas de representação dos dados, que podem ser agrupados em cinco conjuntos principais: Cartesiano, regular, retilíneo, curvilíneo e não estruturados, como na *figura 1.1*.

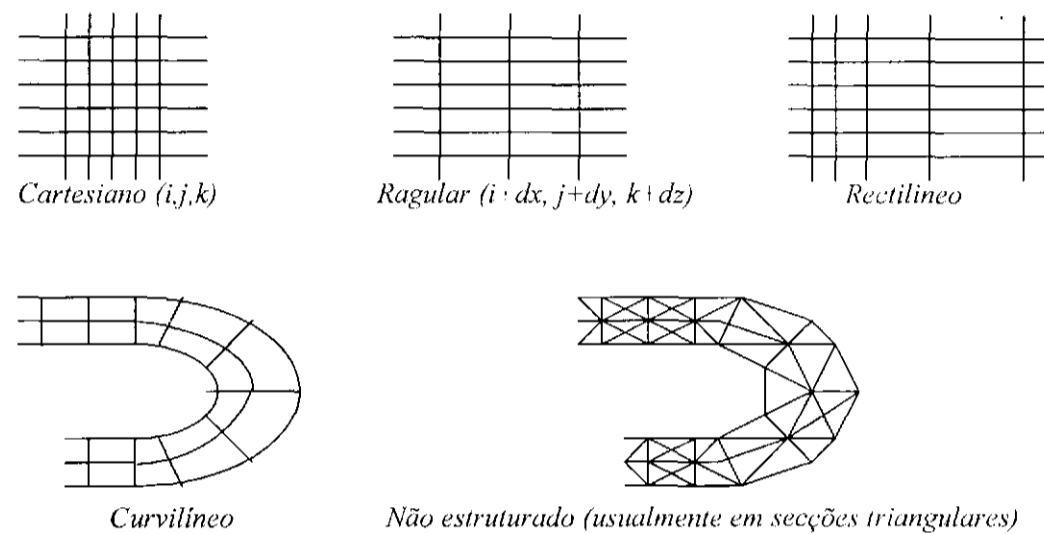


Figura 1.1. Representação dos grupos de tipos de dados [SM97].

Para serem úteis, as técnicas de Visualização Volumétrica têm que tirar proveito de representações apropriadas para cada tipo de manipulação de dados. Assim, a questão dos modelos de representação (*voxels* ou células) e do armazenamento dos dados também é um fator importante durante a visualização.

A forma de representação tradicional dos dados é mediante uma malha regular de *voxels* (Cartesiana). Muitos trabalhos descritos na literatura empregam essa estratégia [WCAR90].

As malhas curvilíneas [WCAR90, MHK95, RW92, Ch92] são também muito comuns por serem malhas estruturadas que preservam a mesma estrutura topológica de uma malha regular.

Mas quando temos problemas de representação onde existem graus de resolução variáveis em diferentes regiões do espaço, a utilização de malhas regulares não é viável, já que os *voxels*

teriam que ser suficientemente pequenos para representar os mínimos detalhes que podem caracterizar a imagem.

Métodos que geram malhas adaptativas de alta qualidade estão adotando malhas irregulares não estruturadas na representação de volumes de dados.

Existem variados problemas relacionados à manipulação de malhas não estruturadas. Um dos principais refere-se à dificuldade de implementar, nesse tipo de representação, algoritmos de visualização volumétrica.

Dado que grande parte dos algoritmos de visualização trabalha com malhas regulares, geralmente se utiliza uma etapa de pré-processamento para reamostrar os dados irregulares em um novo conjunto de dados regular que possam ser mais facilmente visualizados [Fr94].

Outro problema das malhas não estruturadas está no fato de que o tamanho das representações tende a ser maior que no caso regular, pois as células são decompostas em polígonos. Para propósitos de *rendering*, malhas compostas de células convexas são usualmente empregadas. Em geral, as malhas mais convenientes são as malhas tetraedrais e hexaedrais. Por ser de nosso interesse, quando nos referirmos a malhas não estruturadas estaremos referindo-nos à malha tetraedral, não só porque o tetraedro é um tipo popular de volume, mas especialmente pelo fato que qualquer volume pode ser decomposto em tetraedros [E192].

Neste trabalho de pesquisa foram investigados os principais problemas relacionados ao *rendering* para cumprir com o objetivo de elaborar e implementar novos recursos de interação e visualização para exploração de volumes não estruturados. Com o objetivo de construir um sistema de *rendering* volumétrico, utiliza-se o *Ray Casting* como algoritmo base para construção de um algoritmo que faz uso dos métodos de *Rendering Superficial* (RS) e de *Rendering Volumétrico Direto* (RVD). Essa função deu origem à um Novo Algoritmo Híbrido de Visualização Volumétrica para Malhas Não Estruturadas.

Assim, desenvolveu-se uma possível solução para conseguir uma interação mais efetiva durante a visualização dos dados, tirando vantagem tanto das características das malhas não estruturadas, como da informação volumétrica. O algoritmo híbrido proposto nesse trabalho é uma alternativa à maioria dos algoritmos que estão presentes na literatura, com a finalidade de possibilitar uma visualização em tempo real ao pré-processar parte dos dados, mantendo uma boa qualidade da imagem resultante.

Contudo, não faz parte deste trabalho investigar ou analisar os procedimentos de extração das malhas não estruturadas que serão utilizadas, ou a resolução das mesmas, muito menos avaliar os resultados que apresentam. Assume-se que a malha tetraedral foi obtida com ajuda de algoritmos especializados e que modelam os objetos a serem visualizados da forma mais aproximada possível.

Este trabalho está organizado da seguinte forma. No *capítulo 2* é apresentado um resumo dos principais algoritmos de visualização volumétrica, assim como as definições relacionadas a RS e RVD. No *capítulo 3* são apresentadas as diferentes noções do termo “algoritmo híbrido” junto com os trabalhos desenvolvidos nesses contextos. No *capítulo 4* é descrita a técnica de *Ray Casting* juntamente com as técnicas de Visualização Volumétrica que utilizam ela. No *capítulo 5* apresenta-se a técnica híbrida para a visualização de volumes não estruturados desenvolvida neste trabalho de mestrado. O *capítulo 6* apresenta as conclusões deste trabalho e o *capítulo 7* as sugestões para trabalhos futuros.

CAPITULO 2

Visualização Volumétrica

2.1. Considerações Iniciais

Visualização Volumétrica é o processo de projetar um conjunto de dados multidimensionais em uma imagem bidimensional plana [E192]. O objetivo da Visualização Volumétrica é prover mecanismos para visualizar o conteúdo dos conjuntos de dados volumétricos e manipular estas estruturas volumosas, complexas e dinâmica. A Visualização Volumétrica investiga estruturas para representar os dados e técnicas de *rendering* para sua visualização.

Freqüentemente, o conjunto de dados a ser visualizado é definido como uma malha tridimensional com um ou mais valores escalares e, possivelmente, um ou mais valores vetoriais em cada elemento da malha. A forma de representação destas malhas varia desde malhas regulares, tipicamente formadas por unidades de volume cúbicas denominadas *voxels*, até malhas irregulares, formadas por células que apresentam características mais gerais que os *voxels*. Tais células podem adotar diferentes formas geométricas, entre elas a tetraedral, que é uma das mais utilizadas atualmente.

Os métodos de Visualização Volumétrica sofreram um grande avanço após o surgimento de técnicas não invasivas para captura de informações internas, como a ressonância magnética, tomografia computadorizada e ultra-sonografia.

Os métodos de visualização de dados volumétricos podem ser divididos, basicamente, em duas categorias: *Rendering* de Superfícies (*surface rendering*), e o *Rendering* Volumétrico Direto (*direct volume rendering*).

A seguir apresenta-se uma descrição das formas de representação de volume e as principais técnicas de visualização para *Rendering* de Superfície e *Rendering* Volumétrico Direto.

2.2. Descrição do Volume

Como foi mencionado antes, um dos objetivos da Visualização Volumétrica é fornecer estruturas para representar objetos volumétricos, e assim, possibilitar uma melhor visualização do e manipulação da informação presentes em tais objetos.

Os dados volumétricos tipicamente são constituídos de um conjunto de elementos da forma (x, y, z, v) , onde o valor v representa alguma propriedade do dado em uma posição do espaço (x, y, z) . Se o valor é simplesmente 0 ou 1, tal que 0 indica ausência do objeto e 1 indica presença do objeto, então os dados são referenciados como binários. Os dados também podem ser escalares ou multivalentes, isto é, o valor v pode ser escalar e representar alguma propriedade mensurável, como por exemplo, cor, densidade, calor ou pressão; ou uma grandeza vetorial, como velocidade em cada ponto do espaço.

Os dados volumétricos são obtidos realizando uma amostragem de dados no espaço. A amostragem gera um conjunto de dados (malha ou reticulado) que pode ser: regular, curvilíneo, irregular ou disperso [SK90] dependendo da função utilizada para determinar a posição onde as amostras são tomadas. Se a amostragem é feita em intervalos regulares ao longo de três eixos ortogonais, obtém-se uma malha regular. Nesse caso, pode-se admitir que o espaçamento entre as amostras seja constante, podendo existir três diferentes constantes para os três eixos. Se a amostragem é feita em pontos aleatórios no espaço, tem-se uma malha irregular. Uma classificação bastante utilizada para descrever a estrutura espacial dos dados volumétricos pode ser encontrada em [SK90, PSG99].

Utilizam-se basicamente dois tipos de representação para dados volumétricos, o *voxel* e a célula. As duas representações somam vantagens e desvantagens nos algoritmos que adotam uma ou outra abordagem [E192].

A representação mediante *voxels* define uma área ao redor de um ponto amostrado que tem o mesmo valor do ponto. Um *voxel* pode então ser definido como uma região hexaedral com propriedades constantes ao redor de um ponto amostrado (*figura 2.1a*). A vantagem da representação por *voxel* é que os dados volumétricos conhecidos são únicos, ou seja, nenhum valor interpolado é calculado entre os dados amostrados. Uma desvantagem do *voxel* é a dificuldade que é imposta por ele na representação de detalhes finos.

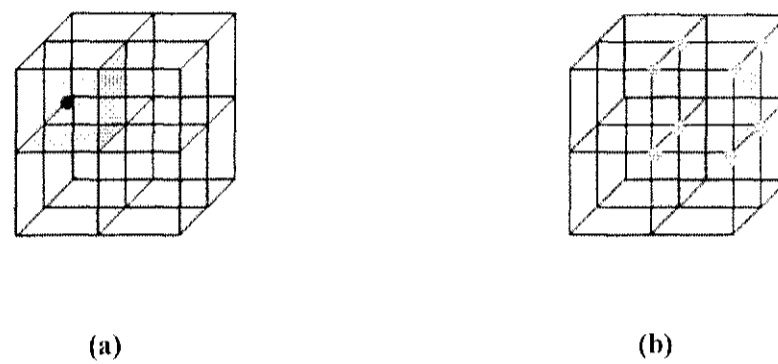


Figura 2.1. Unidades de Representação Volumétrica: a) *voxel* e, b) *célula*. Os pontos pretos representam os dados volumétricos.

A representação mediante *células* considera o volume como uma coleção de poliedros, nos quais os vértices são pontos amostrados com valores variando entre tais pontos. Esta técnica tenta estimar os valores dos pontos que estão dentro da célula mediante interpolação dos valores amostrados nos vértices, como se observa na *figura 2.1b*. Funções de interpolação típicas é a de ordem zero, *figura 2.2*, que é uma função de “vizinhança” (*nearest-neighbor*) [Ka97], e a interpolação de ordem superior, geralmente as trilineares e tricúbicas, *figura 2.3*. As imagens geradas usando células são mais suaves que as geradas mediante *voxels*, mas a exatidão na localização de estruturas internas fica prejudicada com essa abordagem.

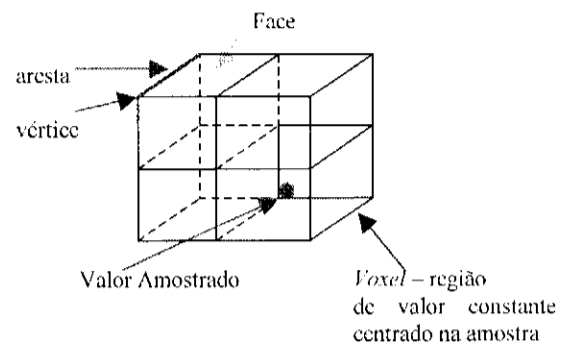


Figura 2.2. Interpolação de ordem zero.

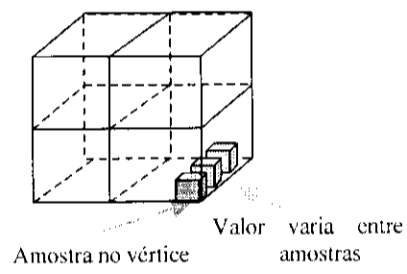


Figura 2.3. Interpolações de ordem superior

Diferente do *voxel*, a célula pode apresentar diferentes formatos polidrais de representação, *figura 2.4*. Casos particulares são os tetraedros.

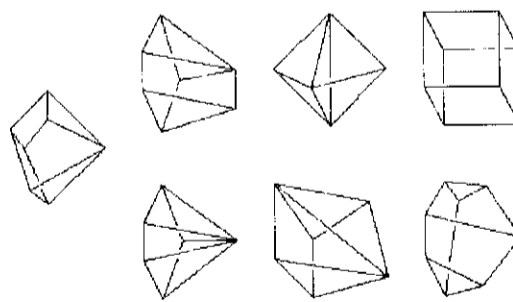


Figura 2.4. Diferentes formas polidrais adotadas como células.

Tetraedros também são chamados de células simpliciais ou simplexos. Os conjuntos de simplexos, também conhecidos como complexos simpliciais (conjuntos de tetraedros em R^3), são muito utilizados na Visualização Volumétrica [CMS97, SM97, CMPS97, WMS98, CKMM99], pois: a maior parte das células podem ser decompostas em complexos simpliciais; o *rendering* pode ser feito mais facilmente para os complexos simpliciais; a extração de iso-superfícies a partir de complexos simpliciais evita ambigüidades presentes em outras representações; a maioria dos algoritmos de *rendering* são mais simples (e rápidos) de descrever e implementar para complexos simpliciais.

Além das vantagens acima mencionadas, as malhas tetraedrais são adequadas para modelar dados em qualquer dimensão, são bases adequadas para muitas técnicas de interpolação, o projeto de estruturas de dados é simplificado, o tratamento de casos degenerados é mais simples. Outras razões ainda para adotar complexos simpliciais na Visualização Volumétrica são apresentadas em [CMS97].

2.3. Classificação dos Algoritmos de Visualização Volumétrica

Os algoritmos fundamentais de Visualização Volumétrica, segundo as técnicas de *rendering* que utilizam, pertencem à uma das duas categorias básicas:

1. *Rendering Superficial (RS) - Surface Rendering.*
2. *Rendering Volumétrico Direto (RVD) - Direct Volume Rendering.*

O *Rendering Superficial*, também conhecido como reconstrução de superfícies mediante algoritmos, é geralmente feito através de aproximações poliedrais de iso-superfícies extraídas do volume. O *Rendering Volumétrico Direto* utiliza a projeção direta dos dados que compõem o volume para gerar uma visualização. Uma explicação mais ampla é oferecida nas próximas seções deste capítulo.

Os algoritmos de Visualização Volumétrica requerem, em sua maioria, uma série de passos comuns [Ka91] que são apresentados na *figura 2.5*. Esta série de passos básicos é conhecida como o *Pipeline* de Visualização. O primeiro passo consiste na aquisição do dado volumétrico, que normalmente é feito utilizando equipamentos de sensoriamento apropriados ou através de simulações. O segundo é a classificação do dado volumétrico bruto, cuja finalidade principal é

identificar estruturas internas do volume através do uso adequado de cores e opacidades: este passo normalmente é feito com o auxílio do usuário para a eleição de limiares (*thresholds*) e será discutido com maiores detalhes no *capítulo 4*. Em seguida, o volume classificado é iluminado a fim de realçar a forma tridimensional do volume e facilitar a interpretação pelo usuário. O último passo consiste na projeção em um plano de visualização do volume classificado e iluminado, gerando assim uma imagem bidimensional que o usuário visualizará. Este último passo é o que mais diferencia os algoritmos.

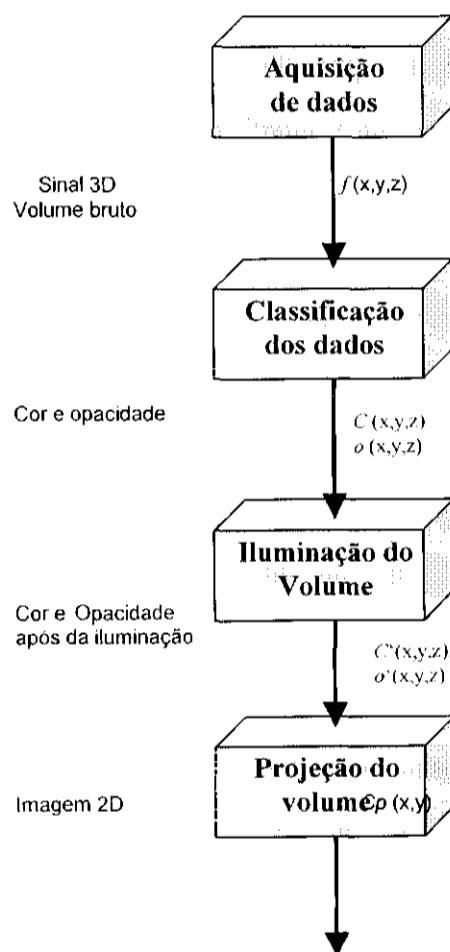


Figura 2.5: Pipeline de Visualização Volumétrica.

O interessante ao estudar este *pipeline* é perceber que cada etapa está profundamente relacionada com as anteriores. Então, qualquer modificação no processo de visualização torna necessário revisitar as etapas posteriores. Por exemplo, se o usuário mudar uma das funções de transferência utilizadas na classificação, as etapas de iluminação e projeção precisam ser novamente executadas.

2.4. Algoritmos de *Rendering* Superficial (RS)

Os algoritmos de RS utilizam primitivas geométricas para gerar um modelo do objeto, a partir do qual, o volume foi obtido. Neste caso, são geradas apenas as superfícies que limitam regiões dos objetos, sendo que as técnicas mais comuns são: extração de iso-superfícies [I.C87, CLLC88, Le90b, NH91, HL79, CMS97, CMPS97] e os métodos de reconstrução a partir de contornos [Ke90, EPO91, JC94, FKU97, Ka98, BCL99, KSS00].

Tipicamente, a extração de iso-superfícies aproxima a fronteira das regiões de interesse contidas no volume, com primitivas geométricas (geralmente planas). Uma superfície pode ser definida mediante distintas técnicas, como por exemplo, aplicando uma função de segmentação binária $B(v)$ nos dados volumétricos, isto é, se $B(v) = 1$, o valor v é considerado como parte do objeto e se $B(v) = 0$, o valor é considerado como parte do fundo (*background*). A superfície é a região onde $B(v)$ muda de valor (de 0 para 1).

A fim de decidir quando $B(v) = 0$ ou $B(v) = 1$, o usuário deve eleger um valor de limiar (*threshold*), e então as primitivas geométricas aproximam automaticamente as fronteiras do volume que correspondem a esse limiar. As células cujos valores nos vértices estão todos acima do limiar (estão dentro) ou todos abaixo (estão fora) são descartadas e não têm efeito na imagem final. Entre as vantagens dos algoritmos de RS observa-se que geralmente são mais simples de calcular e mais rápidos que os algoritmos RVD, pois o RS atravessa o volume uma única vez para extrair as superfícies.

O RS também apresenta desvantagens. Já que é utilizada somente uma representação da superfície, muita informação contida nos dados é perdida durante o processo de *rendering*. Dois casos exemplificam claramente este ponto. Primeiro, no caso das tomografias computadorizadas, informações úteis não estão só nas superfícies das imagens, mas também no interior dos dados. O segundo caso acontece com os fenômenos amorfos como as nuvens, fumaça, e fogo, os quais não

podem ser adequadamente representados utilizando superfícies e requerem uma representação volumétrica para que possam ser apresentados adequadamente.

Outra consideração é que as primitivas geométricas podem aproximar as superfícies contidas nos dados originais, mas obter aproximações adequadas pode levar a uma quantidade excessiva de primitivas geométricas.

Alguns outros problemas são a introdução ocasional de falsos positivos, isto é, formas que não existem; ou de falsos negativos, isto é, não levar em conta características dos dados pequenas ou definidas de forma muito simples. Os falsos positivos e negativos podem ser vistos incorretamente pelos usuários como características dos dados [E192].

As principais técnicas de extração por iso-superfície para malhas regulares são: *Cuberille* [HL79] e *Marching Cubes* [CLLC88], já para malhas não estruturadas uma das principais técnicas desenvolvidas foi uma extensão do algoritmo *Marching Cubes* denominada *Marching Tetrahedra* [CMS97]. Dentro do que é a reconstrução a partir de contornos encontramos o algoritmo mais representativo é o *Contour-Connecting* [Ke97]. Estes algoritmos são descritos a seguir.

Cuberille (Opaque Cubes – Cubos Opacos)

Foi criado por Herman [HL79] e melhorado por muitos outros [E192, Sc99b]. Este método de *rendering* por extração de iso-superfícies baseia-se em uma interpretação por cubos dos objetos de uma cena.

O algoritmo consiste de duas etapas: na primeira, o usuário determina um valor de limiar que é utilizado para detectar os elementos que satisfazem esse critério. O volume (no espaço do objeto) é percorrido, procurando células cujos vértices apresentem valores próximos (acima ou abaixo) do valor de limiar (*figura 2.6*). Para cada uma dessas células gera-se seis polígonos, um para cada face da célula. A segunda etapa é o *rendering* do modelo 3D. Nessa etapa, as descrições geométricas dos polígonos gerados na etapa anterior são passadas para um visualizador de superfícies que gera a imagem. Se for determinado mais de um valor de limiar, cada conjunto correspondente de polígonos pode ser visualizado com uma cor e opacidade diferentes.

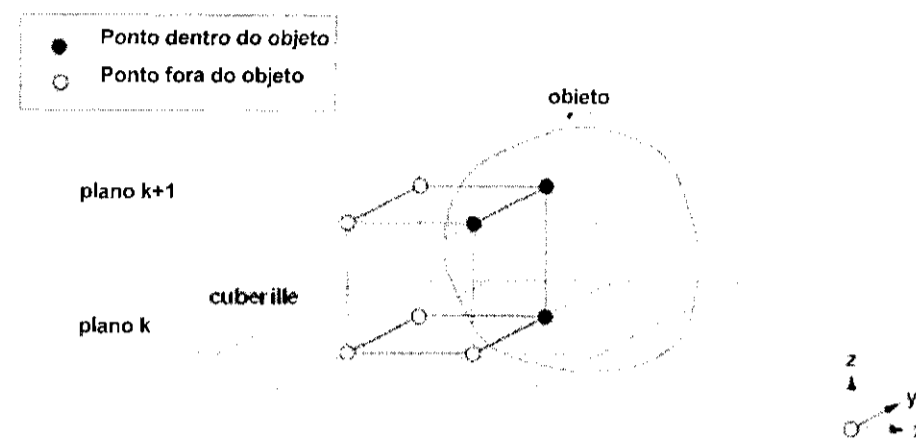


Figura 2.6. Rendering Superficial - faces das células [Sc99b].

Os cubos opacos ou semi-opacos usados para representar os contornos das superfícies de igual valor fazem com que a superfície pareça formada por blocos na imagem visualizada. A aparência de blocos da iso-superfície aproximada pode ser reduzida usando sombreado por gradientes no passo de visualização.

Este método apresenta a maior parte das deficiências dos métodos RS [E192] antes mencionados, especialmente na visualização de pequenas características nos dados. Dentro de suas vantagens tem-se que ele isola o objeto de interesse das estruturas vizinhas e calcula o volume do objeto. Além disso, é fácil de implementar, e é relativamente rápido. O primeiro passo deste algoritmo pode ser paralelizado facilmente, pois todas as células podem ser testadas e os polígonos constituídos de forma independente.

Marching Cubes

O algoritmo de RS mais popular e muito empregado atualmente é o *Marching Cubes* de Lorensen e Cline [LC87, CLLC88]. Este algoritmo também utiliza um valor de limiar determinado para pesquisar quais células contribuem na formação da superfície. Além disso, a função de interpolação geralmente utilizada é contínua o que possibilita a geração de superfícies mais suaves.

Este algoritmo foi amplamente implementado e trabalha da seguinte maneira: a partir de dados amostrados regularmente, percorre-se cada célula (cubo) formada por quatro pontos adjacentes localizando-se as que contêm as iso-superfícies. Triângulos são ajustados dentro de cada uma dessas células dando origem a uma aproximação da iso-superfície procurada. A seguir, os

gradientes são calculados, para assegurar uma imagem de alta qualidade. Finalmente, os triângulos são passados para um visualizador de polígonos que os projeta no espaço da imagem [E192].

Este algoritmo considera o número finito de casos de interseção de uma iso-superfície, com a célula hexagonal. Os autores analisaram 256 possíveis configurações de interseção da iso-superfície com uma célula cúbica a partir dos valores dos seus vértices (cada vértice estará acima ou abaixo do iso-valor). Destas 256 configurações, por argumentos de simetria (reflexão, rotação) esses casos foram reduzidos para 15, como mostra a *figura 2.7*. Os pontos pretos (visíveis) e brancos (invisíveis) representam os vértices com valores acima e abaixo do limiar respectivamente.

As células que não contêm vértices com valores em ambos lados do limiar são ignoradas. Os oito vértices do cubo são numerados (1-8) e recebem um valor 1 se estão sobre o limiar, e valor 0 se não estão sobre o limiar. Os oito valores representam posições consecutivas de 8 bits (0-7), formando um byte. Este byte é o índice de uma tabela pré-calculada de interseções dos lados. Uma função de busca de tais interseções devolve 12 valores booleanos, que indicam quais das 12 arestas da célula são intersectadas pela iso-superfície. A posição exata da interseção é determinada mediante interpolação dos valores nos vértices. Assume-se que cada aresta é intersectada uma vez só e, então, no máximo quatro triângulos são suficientes para mostrar o caminho da iso-superfície através da célula. Conjuntos de três pontos, resultantes da interseção das arestas com a superfície, são agrupados para formar triângulos. Os gradientes nos pontos de interseção podem ser calculados interpolando os valores dos gradientes nos extremos da célula. Os gradientes interpolados são armazenados junto com os triângulos e serão utilizados para o posterior sombreamento.

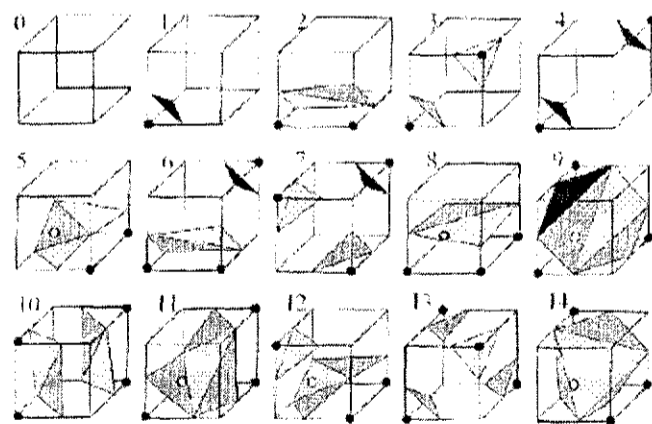


Figura 2.7. *Marching Cubes: as configurações básicas para triangulação de uma única célula [Le90b].*

Os passos deste algoritmo são sintetizados a seguir:

1. Detecção dos vértices cujos valores estão acima do limiar, e cálculo de um índice para uma tabela de intercessão de bordas definindo a configuração dos triângulos dentro da célula.
2. Definição dos vértices dos triângulos por interpolação linear entre os valores dos vértices das células.
3. Pré-cálculo dos gradientes em cada vértice interior da malha, para garantir uma imagem de alta qualidade.
4. Visualização do modelo 3D.

Uma das vantagens desta técnica é que a conectividade dos polígonos em células adjacentes é garantida. Este algoritmo utiliza unicamente informação de conectividade, dado que foi originalmente desenvolvida para volumes provenientes de aplicações médicas, e pode ser aplicado em volumes retilíneos e gerais. *Marching Cubes* não trabalha com valores randômicos porque se perde a informação da conectividade que é vital. Algumas vezes este algoritmo erra na conexão de pontos, resultando falsos positivos e falsos negativos na iso-superfície.

Marching Tetraedra

As situações de conexões de pontos ambíguas, geradas pelo algoritmo *Marching Cubes*, podem ser reduzidas utilizando a técnica de *Marching Tetrahedra* que é basicamente uma versão do *Marching Cubes* para volumes tetraedrais.

A idéia principal do algoritmo *Marching Tetraedra* é atravessar todas as células do conjunto de dados e calcular o corte (*patch*) da iso-superfície para cada célula atravessada por uma iso-superfície (célula ativa). Cada vértice dos dados é classificado como branco, cinza ou preto se o valor associado com o vértice é maior, igual ou menor do que o valor de limiar dado. Esta classificação dos vértices das células tetraedrais podem gerar $3^4 = 81$ configurações diferentes de interseção [CMS97]. Explorando as simetrias, obtém-se seis classes principais de interseção como mostradas na *figura 2.8*.

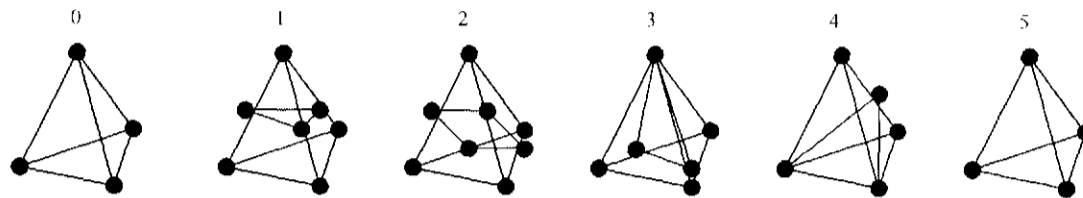


Figura 2.8. Algoritmo *Marching Tetraedra*: configuração dos vértices e dos cortes (*patches*) correspondentes às iso-superfícies [CMS97].

Uma vez que a classe de cada tetraedro foi identificada, a posição dos vértices da iso-superfície e as normais associadas são calculadas mediante interpolação linear.

Note que, se uma iso-superfície passa exatamente através dos vértices de algum tetraedro, então, uma análise de vizinhança deve ser feita para evitar a aparição de triângulos nulos ou duplicados. Quando surge uma conexão ambígua de pontos, cada célula deve ser dividida em cinco [CMPS97], seis ou 24 tetraedros e realizando-se testes de interseção de lados, baseados em tabelas, junto com os tetraedros. Dado que um tetraedro possui quatro lados, dois triângulos serão suficientes para mostrar a iso-superfície dentro da célula tetraédrica. Em geral, este algoritmo gera mais triângulos que o anterior (*Marching Cubes*), e requer maior processamento e memória. Um método alternativo para reduzir a questão de pontos ambiguos é descrito em Nielson [NH91].

Estas técnicas apresentam as mesmas vantagens e desvantagens (em certo grau) mencionadas na definição do RS. Da mesma forma que o método *Opaque Cube*, este método pode ser paralelizado no nível das células.

Conexão de Contornos (Contour-Connecting)

Um dos primeiros métodos para produzir imagens 3D de volumes gerados a partir de seqüências de imagens foi a Conexão de Contornos. A idéia básica do algoritmo é traçar contornos fechados em cada imagem para, então, conectar os contornos em níveis adjacentes [FKN80, EPO91, Ke97].

Contour-connecting é um método RS que inicia operando em cada fatia de dados

individualmente. Depois que o usuário especifica um valor de limiar, deve-se achar, para cada fatia, um contorno fechado que conecte os pontos com esse valor de limiar. Nos primeiros algoritmos dessa classe, o passo de desenhar os contornos era feito manualmente. As técnicas de processamento de imagens proporcionaram métodos para gerar os contornos automaticamente, mas a intervenção humana em algumas ocasiões ainda é necessária.

Uma vez determinados os contornos, uma triangulação conectando as curvas em fatias adjacentes deve ser gerada. Keepel reduz este problema a um caminho em grafo dirigido usando uma estratégia de otimização [Ke97]. Posteriormente, Fuchs e outros autores especificaram o problema como um caminho de custo mínimo em um grafo toroidal dirigido [FKN80].

O último passo, como nos casos anteriores, é passar os triângulos a um visualizador de superfícies. Este método apresenta vários problemas que vão desde a extração dos contornos por meio de técnicas de processamento de imagem (a segmentação) até a construção de uma triangulação apropriada. A *figura 2.9* exemplifica parte da idéia deste método.

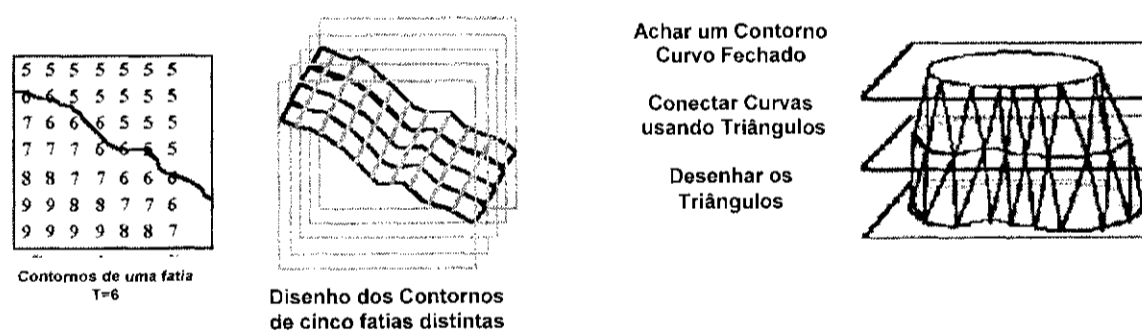


Figura 2.9. Rendering Superficial por triangulação de contornos [Sc99b].

Note que o algoritmo é paralelizável, dado que a triangulação das fatias adjacentes não depende do resultado da triangulação de outro par.

2.5. Algoritmos de *Rendering* Volumétrico Direto (RVD)

Os métodos de RVD são capazes de preservar as informações volumétricas na visualização.

fato que tem levado muitos pesquisadores a investir neste tipo de estratégia [Bo88, BCL96, NMS00]. Estas técnicas objetivam criar uma imagem diretamente a partir dos dados volumétricos, sem o auxílio de primitivas geométricas [Ka98]. As técnicas de RVD se tornaram muito populares por evitar a necessidade de fazer um grande pré-processamento dos dados e por serem relativamente fáceis de implementar para o caso de volumes regulares.

A forma mais simples (e ineficiente) de implementar uma técnica de RVD é utilizando a técnica do *Z-Buffer*. A idéia é atravessar o volume inteiro considerando cada *voxel* como um ponto 3D que é transformado, utilizando uma matriz de projeção em um *Z-Buffer*, e desenhado na tela. A ineficácia deste método deve-se às multiplicações *vetor x matriz* que são de ordem N^3 . Este método produz imagens de menor qualidade porque não suporta a composição de materiais semitransparentes, o que se deve à ordem arbitrária na qual os *voxels* são transformados. Esta técnica é uma das mais simples e foi implementada para malhas regulares conformadas por *voxels*. Posteriormente foram desenvolvidas técnicas de RVD mais eficientes tanto para malhas regulares [KS86, OUT85, Gu88, HM90, NSG90, Gi90, Le90b, Le90d, LH91, YCK92, Sc99a] como para malhas irregulares [Ko90, Ga90, Gi90, WCAR90, Ko92, RW92, Wi92, Ch93, MMS94, Fr94, MIIK95, YRLS96, BKS97, CMS97, SM97, WMS98, CKMM99, CTT00].

A ordem de projeção nos permite dividir as técnicas de RVD basicamente em três categorias: *object-order* [DCH88], *image-order* [YCK92, We94] e *domain volume rendering* [TT84, TL88].

No primeiro caso, *object-order*, a projeção é feita no espaço dos objetos utilizando o mapeamento direto (*forward mapping*) do volume sobre o plano da imagem. Isto significa que os *voxels* são projetados na tela em uma ordem determinada. Os algoritmos “*back-to-front*” (*BTF*) [DCH88] são deste tipo, operando basicamente como o *Z-Buffer*, porém consideram que o *array* de *voxels* está pré-ordenado, de tal forma que uma varredura (*raster*) dos elementos em ordem decrescente ou decrescente com relação à distância do observador possa ser feita. Logo, o algoritmo atravessa o volume a partir dos elementos mais distantes, aproveitando o pré-ordenamento dos *voxels*, evitando a remoção dos *voxels* ocultos (simplesmente desenha o *voxel* atual sobre o anterior). Como exemplo, pode-se citar os algoritmos *Splatting* e *V-Buffer* [LH91, YCK92].

No segundo caso, *image-order*, a projeção é feita considerando o espaço da imagem. Utilizando o mapeamento reverso (*backward mapping*), são lançados raios para cada pixel da imagem em direção ao volume, determinando assim, a cor final do pixel. Com exemplo deste tipo de algoritmo temos o “*front-to-back*” (*FTB*) [GC91]. Este algoritmo pode ser implementado

facilmente utilizando uma estrutura de dados dinâmica, que permita processar somente os *voxels* visíveis, e remover os ocultos de forma eficiente. Para este cálculo a ordem dos materiais é importante. Como exemplo, pode-se citar o algoritmo *Ray casting* [Le90b]. Estes algoritmos projetam o volume diretamente no plano da imagem e são os mais indicados para visualizar volumes que representam objetos amorfos.

Tanto os métodos BTF como FTB permitem representar translucidez mapeando os objetos na tela seguindo a ordem apropriada.

Uma técnica que combina os métodos *image-order* e *object-order* é a transformação de fatias (*shear transformation*) [LL94]. O algoritmo supõe que o volume é dado por fatias e implementa uma transformação da visualização como uma série de transformações das fatias bidimensionais, com reamostragem do conjunto de dados. O resultado da transformação é um volume de dados na memória, alinhado a partir de um determinado ponto de visão. A composição é feita caminhando através da memória na ordem correta. Exemplo desta técnica é o algoritmo *Shear Warp* [L194, La95].

No último caso, *domain volume rendering*, temos as técnicas baseadas no domínio que primeiro transformam os dados de volume para um domínio alternativo, para posteriormente projetar diretamente neste domínio. Temos, como exemplos, as técnicas de projeção no domínio da frequência [Ma93, TL88]. Trabalhar sobre um domínio alternativo, tal como, o domínio da frequência é interessante porque pode reduzir a complexidade do processo de $O(n^3)$ para $O(n^2 \log n)$ [Ma93]. Por exemplo, ao utilizar a teoria de Fourier para realizar *rendering* volumétrico, uma vez que os dados volumétricos foram transformados utilizando Fourier para um domínio de frequência tridimensional, uma imagem (ortográfica) pode ser obtida, para qualquer direção de visão, extraindo uma fatia 2D do espectro 3D obtido pela transformada, para então, aplicar a transformada inversa de Fourier sobre ela (*figura 2.10*), obtendo-se assim a imagem a ser projetada. Esta abordagem é chamada *Rendering Volumétrico de Fourier* [Ma93]. Desafortunadamente, as projeções feitas utilizando esta abordagem não exibem a oclusão que é característica do *rendering* volumétrico convencional, além de apresentar outros problemas como a perda da noção de profundidade (porque, no caso de *voxels*, cada um contribui de igual forma ainda se estiverem a distancias diferentes do observador), e requer de um alto custo de interpolação (do espectro 3D para obter a fatia 2D) [TL88].

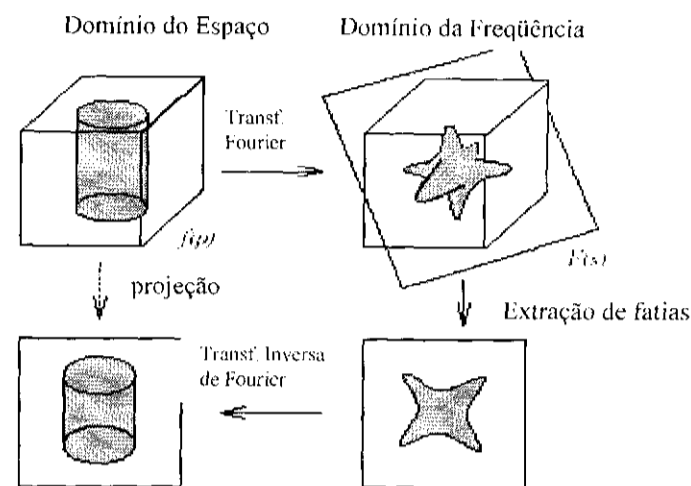


Figura 2.10. Rendering volumétrico utilizando o teorema de projeção de fatias de Fourier.

A grande vantagem dos algoritmos RVD é a alta qualidade da imagem produzida. Em contrapartida, a necessidade de atravessar o volume diversas vezes quando uma imagem é produzida, torna o algoritmo RVD lento, computacionalmente caro, e a interação do usuário com a imagem 3D produzida é crítica, apresentando importantes desafios para os pesquisadores que utilizam esta técnica.

Tipicamente, o *buffer* volumétrico ocupa uma enorme quantidade de memória. Por exemplo, para obter uma resolução média de 512^3 , 2 bytes por *voxel*, o *buffer* volumétrico ocupará 256 Mbytes.

A representação baseada em *voxels* também apresenta um limite na precisão de algumas operações tais como medições de área e volume, pois se baseiam na contagem de *voxels* que pode ser uma aproximação grosseira. Devido à natureza discreta dos dados amostrados durante o *rendering*, um volume de baixa resolução pode apresentar um efeito de serrilhado ou *aliasing* muito alto (artefatos), que podem aparecer até como buracos entre os *voxels*. Além disso, quando um objeto é “voxelizado”, perde-se a informação de geometria. Esta falta de informação traz dificuldades como a necessidade de ajustar algum tipo de primitiva de superfície dentro de uma pequena vizinhança de *voxels* para poder estimar normais (elemento utilizado para calcular os métodos de sombreado, geralmente utilizado nos métodos de visualização).

Embora as malhas cartesianas sejam a forma típica de representar os volumes no RVD, atualmente a tendência é utilizar uma forma de representação mais geral, como são as malhas não estruturadas. Aqui, as operações de *rendering* apresentam novos problemas e ambigüidades. Nas diferentes abordagens referentes ao *rendering* volumétrico de malhas irregulares, estender a operação de amostragem de pontos no lançamento de raios (*Ray Casting*) se apresenta como um desafio para os pesquisadores. A operação de ordenação das células também não é uma operação trivial [Ga90].

A seguir, é apresentada uma visão geral dos algoritmos mais representativos de cada uma destas técnicas de RVD.

Splatting

O algoritmo *Splatting* proposto por Westover [We90], trabalha no espaço dos objetos e procura projetar cada *voxel* do volume no plano da imagem. Essa projeção normalmente é realizada a partir dos *voxels* mais próximos do observador até o mais distante. A idéia central é “arremessar” cada *voxel* sobre o plano da imagem, o que deixa uma “marca” característica do *voxel*, daí o nome do algoritmo.

O primeiro passo do algoritmo é determinar em que ordem o volume será percorrido. Isto é essencial para o correto cálculo da visibilidade, pois a ordem correta de projeção permite acumular adequadamente as opacidades dos *voxels*. Para isso, deve-se escolher os dois eixos do volume mais paralelos ao plano da imagem para formar uma camada de arremesso. Com isso, o plano formado por esses dois eixos será rapidamente projetado e o usuário poderá perceber rapidamente a formação da imagem final. A projeção é realizada fatia por fatia, ou seja, todos os *voxels* de uma determinada camada são projetados antes que a próxima seja processada. Como ocorre nos demais algoritmos, o valor de densidade de cada *voxel* é classificado de acordo com as funções de transferências de cor e opacidade e, a seguir, é iluminado utilizando a técnica de estimativa da normal através do gradiente.

Em seguida, é calculada a contribuição de cada *voxel* no plano da imagem. O algoritmo procura reconstruir um sinal contínuo a partir de um conjunto discreto de dados. Para isto, utiliza-se um filtro de reconstrução (*kernel*) para calcular a região afetada pela projeção de um *voxel* sobre o plano da imagem. A projeção do *kernel* é chamada de *footprint* e, no caso de projeções paralelas, o *footprint* é o mesmo para todos os *voxels*. Isto significa que ele pode ser gerado em uma etapa de

pré-processamento. A extensão do *footprint* está relacionada com o tamanho do volume e a resolução da imagem, ou seja, se a resolução da imagem for maior que o tamanho do volume, a projeção de um único *voxel* pode ocupar vários pixels.

O valor da cor e opacidade de cada *voxel* é composto com os pixels já presentes no *buffer*, levando em consideração a atenuação provocada pela aplicação do *footprint*. Na verdade, o *footprint* é uma tabela que determina como o *voxel* será “arremessado” sobre o plano da imagem. Isso significa que a contribuição do *voxel* é maior no centro de projeção sobre o plano da imagem (pixel central) e menor nos pixels mais afastados. Quando um determinado pixel do plano de projeção acumula opacidade próxima de 1.0, este pixel não precisa mais ser processado.

O algoritmo *Splatting* permite gerar imagens de alta qualidade, porém é muito sensível ao tamanho da tabela de *footprint*. Tabelas pequenas geram imagens com muitos artefatos, enquanto tabelas grandes suavizam demais o volume. Como a projeção é realizada a partir do plano frontal para o posterior, uma vantagem do algoritmo é permitir que o usuário acompanhe o processo de geração da imagem final. Este algoritmo é facilmente paralelizável, pois a projeção de cada *voxel* é realizada de modo independente. Entretanto, a ordem correta de projeção deve ser respeitada.

Uma otimização para o algoritmo de Westover, chamada *Hierarchical Splatting*, foi proposta por Laur e Hanrahan [LH91] e utiliza um “caminhamento hierárquico” nos dados e refinamento progressivo da imagem. Um algoritmo similar, denominado *V-buffer*, foi apresentado por Upson e Keeler [UK88] e tem como principal característica o fato de se basear em células ao invés de *voxels*. O algoritmo de *V-buffer* percorre o interior das células, interpolando os valores dos vértices e projetando cada valor interpolado no plano de visualização.

Ray casting

Uma das técnicas mais utilizadas para a visualização volumétrica é o *Ray casting*. Esta técnica permite a visualização de pequenos detalhes internos ao volume, tem um bom controle de transparência removendo trivialmente as partes escondidas, e visualiza o volume a partir de qualquer direção.

O algoritmo *Ray casting* de Levoy [Le88, Le90a] permite a geração de imagens de alta qualidade através do *rendering* direto, operando no espaço da imagem. A idéia básica do algoritmo é lançar raios na direção de visão atravessando cada pixel da tela (plano de visualização ou plano de

projeção), conforme mostrado na *figura 2.11*.

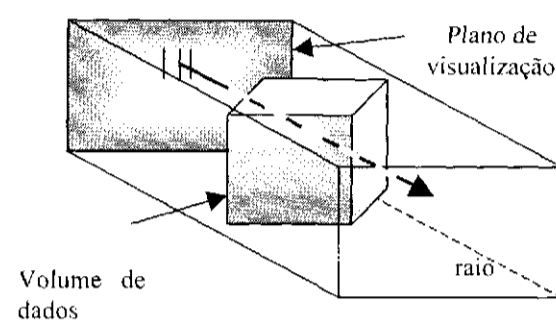


Figura 2.11. Algoritmo de Ray Casting [SGDM95].

Quando a projeção é paralela, a direção destes raios é a da normal a este plano (*figura 2.11*). Para cada raio, integram-se as contribuições dos *voxels* interceptados por ele, determinando assim a cor do pixel. O cálculo da contribuição da cor e da transparência de cada *voxel* baseia-se no valor da densidade e numa estimativa do gradiente da iso-superfície correspondente.

Para o cálculo da contribuição do *voxel* na cor do pixel, utilizam-se modelos de iluminação clássicos. Esta é a etapa de iluminação do *pipeline* de Visualização Volumétrica, aqui a cor de cada *voxel* é modificada utilizando algum modelo de iluminação local, tipicamente Phong. Para isto, o vetor normal em cada *voxel* é calculado através de uma estimativa local do gradiente da função volumétrica, normalmente utilizando diferenças finitas. Além disto, algumas luzes podem ser posicionadas na cena. Esta técnica será melhor explicada no *capítulo 4*, particularmente na *seção 4.4*, por ser parte básica deste trabalho de pesquisa.

Shear-Warp

A idéia principal do algoritmo *Shear-Warp*, apresentado por Lacroute [LL94, La95], é transformar o volume de dados de modo a simplificar a etapa de projeção do *pipeline* de visualização. Isto é através da aplicação de um cisalhamento nas fatias do volume, os dados são

transformados para um sistema de coordenadas intermediário, cuja principal característica é que os raios de visão são perpendiculares às fatias do volume (*figura 2.12*).

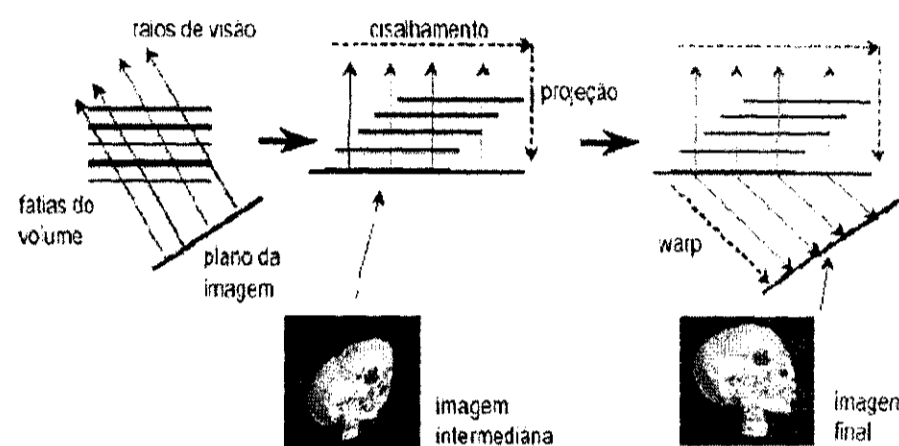


Figura 2.12. Algoritmo Shear-Warp[LL94].

Isso facilita bastante a projeção das fatias, pois os dados volumétricos são acessados na ordem de armazenamento. Note que isso não acontece no algoritmo *Ray Casting*. O cisalhamento é uma transformação geométrica que simplesmente translada as fatias do volume, não sendo, portanto, computacionalmente cara.

A composição e projeção das fatias nesse novo sistema de coordenadas (realizada através do operador *over*) geram uma imagem intermediária distorcida, que precisa ser corrigida posteriormente (através da transformação de *warp*). A principal vantagem desse processo é que as dimensões da imagem intermediária dependem apenas do tamanho do volume de dados e do fator de cisalhamento, não dependendo, portanto, da resolução final da imagem. Como as dimensões do volume de dados são tipicamente menores que a resolução final da imagem, o algoritmo *Shear-Warp* leva uma grande vantagem em relação ao algoritmo *Ray Casting*.

A principal razão para utilizar essa transformação no volume é que as *scanlines* de *voxels* em cada fatia ficam paralelas com as *scanlines* de *pixels* na imagem intermediária, facilitando bastante a projeção das fatias. Após a geração da imagem intermediária distorcida, esta é

transformada na imagem final através da aplicação do *warp*. Esta transformação é realizada em 2D e restaura as dimensões verdadeiras da imagem que será visualizada pelo usuário. De uma forma geral, o ganho de velocidade do algoritmo *Shear-Warp* pode chegar a ser de 5 a 10 vezes em relação ao algoritmo *Ray Casting*, mas só pode ser aplicado diretamente para conjuntos de dados cartesianos.

2.6. Considerações Finais

O objetivo deste capítulo foi apresentar uma visão geral, de forma bastante concisa, das técnicas de Visualização Volumétrica, particularmente das técnicas de *Rendering* de Volumes.

Após uma breve introdução, foi apresentada uma visão global dos principais conceitos de Visualização Volumétrica. Foram então, definidas as terminologias básicas sobre as formas de representação de dados de volume. Em seguida, uma seção foi dedicada à apresentação das diferentes abordagens de *Rendering* de Volumes e seus métodos mais representativos. Foram apresentadas as abordagens de *Rendering* Superficial e de *Rendering* Volumétrico Direto como técnicas capazes de gerar imagens bidimensionais de estruturas externas ou internas de objetos volumétricos. Também foram consideradas as vantagens e desvantagens de ambas abordagens para uma melhor análise das técnicas.

CAPÍTULO 3

Técnicas de Visualização Híbridas

3.1. Considerações Iniciais

Muitas vezes, a forma de visualizar dados volumétricos varia conforme os requerimentos dos usuários e a complexidade dos dados. Isto leva a atualizações constantes dos algoritmos e técnicas de visualização volumétrica empregadas na análise e manipulação de tais dados. Essas atualizações, por sua vez, buscam novas formas de otimização para os métodos de visualização, resultando em algoritmos híbridos, paralelismo, acessibilidade em redes, e visualização em tempo real. Por ser o foco deste trabalho, será apresentada a seguir, uma visão geral do que se entende hoje como algoritmo híbrido de visualização volumétrica. Como a conceituação do termo “híbrido” é muito ampla, neste capítulo são apontadas algumas das tendências adotadas na hibridização de técnicas de visualização.

3.2. Os Algoritmos de *Rendering* Híbrido

Nesse trabalho, o termo “algoritmo híbrido de visualização” se refere a algoritmos que de alguma forma, fazem uso de duas ou mais estratégias de visualização em um único contexto. Poucos algoritmos descritos na literatura fazem uso de técnicas volumétricas diretas distintas como, por exemplo, misturar técnicas de projeção com técnicas de *Ray Casting*. Um exemplo característico deste caso de hibridização é o algoritmo *Shear-Warp* [Ko92]. Alguns algoritmos

realizam simultaneamente o *rendering* volumétrico com o *rendering* de superfícies, permitindo que objetos definidos geometricamente apareçam na mesma imagem com campos de dados amostrados [Le90c]. Outros exemplos utilizam o *rendering* de objetos definidos geometricamente, e de volume com uma operação de composição (segundo uma ordem de profundidade – *depth order*) para formar a imagem final [LFPR90].

Também encontramos algoritmos que utilizam arquiteturas especializadas para desenvolver técnicas paralelas, tal qual a técnica de hibridização descrita em [SLFP00]. Nela, apresenta-se um algoritmo de *rendering* híbrido paralelo que combina duas estratégias de ordenação (*Sort-first* e *Sort-last*), sendo que este algoritmo é dependente do ponto de visão e particiona dinamicamente, tanto o espaço bidimensional da tela, como os polígonos tridimensionais, para obter um balanceamento da carga no *rendering* entre os computadores de um *cluster*.

Após o levantamento bibliográfico notamos que os algoritmos híbridos tendem a ser de três tipos:

- Hibridização segundo o tipo de *rendering*;
- Hibridização segundo o domínio do volume;
- Hibridização segundo a arquitetura de computação.

3.2.1. Hibridização Segundo o Tipo de *Rendering*

Nesta categoria, encontram-se aqueles algoritmos que misturam a visualização de um volume de dados com a visualização de um conjunto de polígonos, procurando resolver problemas envolvendo não só a informação do volume de dados, mais também informações espaciais específicas, que só podem ser proporcionadas pelos objetos definidos geometricamente.

Um exemplo disto está em [Le90c], onde se apresenta a idéia de um algoritmo que mistura um volume de dados com um conjunto de polígonos, de modo que apareçam juntos em uma mesma imagem. Outros exemplos desta categoria são os algoritmos que realizam o *rendering* de objetos definidos geometricamente e o *rendering* volumétrico separadamente, e

através de um pós-processamento, que utiliza a informação de profundidade se gera a imagem final [Le90d, Le90e].

Um caso concreto deste tipo de hibridação é o *Ray Tracing* híbrido apresentado por Levoy [Le90c]. Este algoritmo considera o problema de estender o *rendering* volumétrico para manipular objetos definidos poligonalmente. Na solução proposta por ele, os raios são lançados simultaneamente no conjunto de polígonos e no arranjo de dados volumétricos. Amostras de cada um são extraídas a intervalos igualmente espaçados no caminho dos raios, e as cores e opacidades resultantes são compostas, seguindo a ordem da profundidade. Para evitar o efeito de serrilhado das arestas dos polígonos, foram utilizadas formas de superamostragem seletiva. Para evitar erros na visibilidade, nos pontos de interseção entre os polígonos e o volume, foi dado um tratamento especial para as amostras de volume que se encontraram imediatamente a frente ou atrás dos polígonos.

Outro caso também apresentado por Levoy [Le88] é o algoritmo para a representação de superfícies. Este algoritmo explora a aplicação de técnicas de *rendering* de volumes para visualizar superfícies contidas em volumes. Este algoritmo considera a iluminação, filtragem, e conversão (na varredura) de cada polígono do objeto geométrico, na resolução do objeto volumétrico, na geração de um segundo volume de dados contendo a geometria. Os dois volumes são então combinados utilizando *volume matting*. Cuidado especial é tomado com a ordem na qual os polígonos são convertidos durante a varredura. O volume resultante é então renderizado usando o algoritmo de *rendering* volumétrico direto. Os cálculos de iluminação e classificação são feitos independentemente, para assegurar que regiões pequenas ou características pobremente definidas não sejam perdidas. As cores e opacidades resultantes são compostas de trás para frente na trajetória dos raios, para formar uma imagem. Esta técnica é simples e rápida, mas a representação de superfícies apresenta silhuetas suaves e alguns outros artefatos de serrilhado.

Uma adaptação do algoritmo *Splatting*, para tratar superfícies, pode ser encontrado no trabalho de Tost et al. [TPN93]. O algoritmo considera a classificação de todas as superfícies dentro do volume através da construção de uma *Octree* do modelo de superfícies. A representação mediante *Octrees* [Ba00] permite a enumeração hierárquica do espaço, o que possibilita um rápido atravessamento de regiões uniformes ou vazias. Tost [TPN93] utiliza uma variação desta estrutura, marcando nós como pretos e brancos para diferenciar os espaços que

contém ou não um objeto, cinza para identificar nós não terminais (subdivisão), e acrescenta o nó "Face" que pode ser cortado por um plano (figura 3.1).

A *Octree* é construída e percorrida durante a visualização da cena. A construção da *Octree* pode também ser feita em uma etapa de pré-processamento, podendo ser reutilizada caso a cena seja visualizada a partir de uma nova posição do observador. A idéia central do algoritmo é percorrer simultaneamente a *Octree* a partir de sua raiz (representado toda a cena) e o volume a partir do ponto mais distante do observador para o mais próximo (de trás para frente). A cada nó visitado, a geometria contida em seu interior é classificada e as seguintes situações podem ocorrer: a) o nó contém parcialmente um objeto geométrico (nó cinza) possuindo outros nós filhos a serem visitados; b) o nó contém nenhum objeto geométrico (nó branco) e todos seus *voxels* são projetados no plano de imagem; c) o nó contém algum objeto geométrico (nó preto) indicando que o objeto está cheio ou vazio, podendo ser projetado ou tratado como um nó branco; d) o nó contém uma face de um objeto geométrico (nó face) e todos seus *voxels* são visitados de trás para frente, e tratados como nós pretos ou brancos.

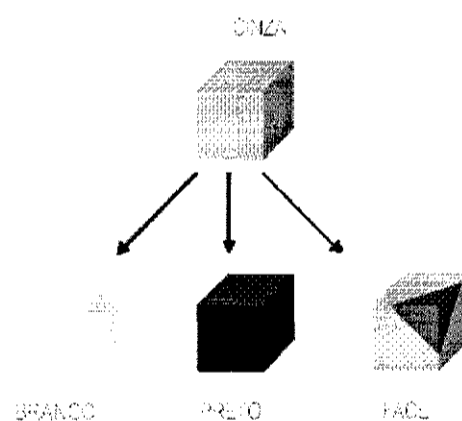


Figura 3.1. Estrutura de dados "Face Octree" [CV00].

No algoritmo híbrido apresentado por Schmidt et al. [SGC99], integram-se os algoritmos *Shear-Warp* e *Z-Buffer*. Este algoritmo procura reduzir o efeito de serrilhado introduzido durante o processo de amostragem das superfícies. Este problema ocorre quando a

resolução do volume é menor que a resolução da imagem visualizada pelo usuário. A idéia principal do algoritmo é que, tanto as fatias do volume, como as superfícies poligonais, sofrem a transformação de *shear*, levando a duas imagens intermediárias independentes que logo são compostas fatia a fatia, como mostrado na *figura 3.2*. A seguir, a transformação de *warp* é aplicada na imagem composta, obtendo-se a imagem final. O problema de serrilhado é resolvido modificando a transformação de *warp* de modo a mapear um pixel da imagem intermediária, em um pixel na imagem final, o que significa que a imagem intermediária deve ser multiplicada por dois fatores M e N.

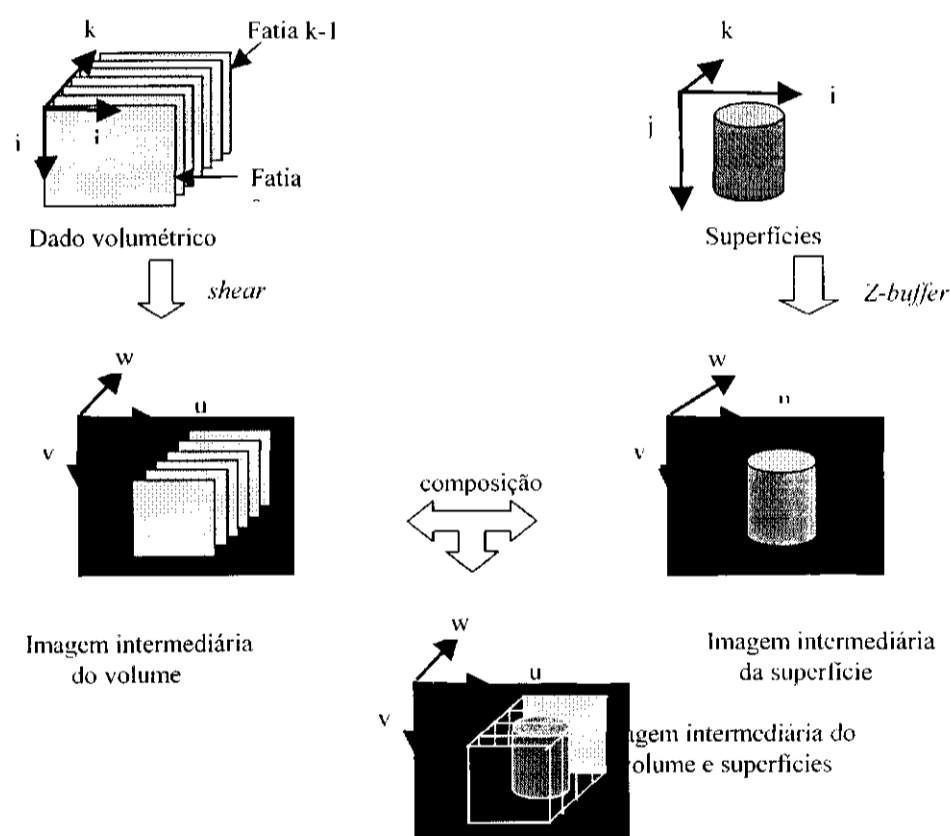


Figura 3.2. Shear-Warp Híbrido [SGC99]

São três as variações deste algoritmo propostas pelos autores, e estas variações estão baseadas na diferença da resolução utilizada para as imagens intermediárias: a) composição em

baixa resolução, em que as imagens intermediárias, tanto do volume, quanto das superfícies, são construídas em baixa resolução, podendo ocasionar um efeito de serrilhado na imagem final; b) composição em alta resolução, na qual as duas imagens intermediárias são construídas em alta resolução, o que exige compor toda a cena em alta resolução; c) composição em resolução dual, em que a imagem intermediária das superfícies é construída em alta resolução, e a imagem do volume pode ser construída em baixa ou alta resolução, dependendo da influência de superfícies, evitando a subdivisão dos pixels quando não há necessidade.

A construção da imagem intermediária das superfícies é baseada no algoritmo *Z-Buffer* padrão, que pode ser realizada por *hardware* específico. O resultado final é um *buffer* de cor e outro de profundidade, com informação da profundidade da superfície mais próxima. A composição é feita fatia a fatia, junto com a construção da imagem intermediária do algoritmo *Shear-Warp* padrão. Após processar cada fatia, os algoritmos contabilizam a contribuição das superfícies, levando em consideração todas as superfícies existentes entre as fatias.

Uma característica da integração do algoritmo *Shear-Warp* e *Z-Buffer* é a dificuldade de implementar transparência das superfícies. Para se conseguir transparência, é necessário, além dos *buffers* de cor e profundidade, um *buffer* de opacidade.

Uma outra variação híbrida deste algoritmo é apresentada por Zakaria e Saman [ZS99]. Este algoritmo mistura o *Shear-Warp* com a estrutura *Zlist-Buffer* proposta por eles. Neste algoritmo, o *Z-Buffer* armazena uma lista de todos os valores de *z* em cada pixel. Portanto, durante a etapa de composição, pode-se levar em consideração a contribuição de todas as superfícies. No trabalho de Schmidt [SGC99] foi implementado um algoritmo que permite até cinco níveis de transparência, utilizando uma estrutura semelhante ao *Zlist-Buffer*.

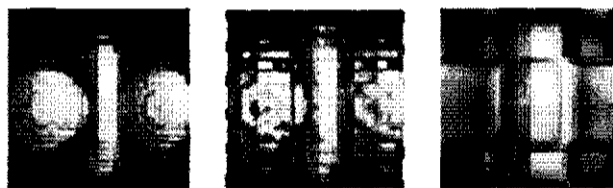
3.2.2. Hibridização Segundo o Domínio do Volume

Estes algoritmos hibridizam técnicas volumétricas baseadas em diferentes espaços de visualização. Um primeiro caso mistura técnicas de projeção a partir do domínio dos objetos com técnicas de projeção a partir do domínio da imagem, um exemplo característico é o próprio algoritmo *Shear-Warp* [LL94], que transforma um volume de dados para simplificar a etapa de projeção do *pipeline* de visualização. Isto é obtido por meio da aplicação de uma translação nas fatias do volume, transformando os dados em um sistema intermediário de coordenadas, onde

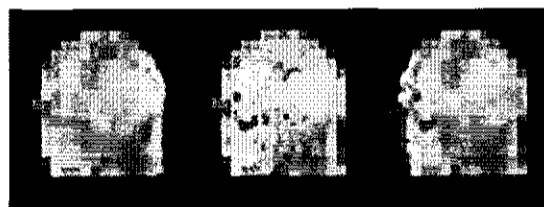
os raios de visão são perpendiculares aos cortes do volume gerando imagens distorcidas que logo são corrigidas para a visualização final (para maiores detalhes desta técnica, vide seção 2.5).

Outros exemplos destes métodos de *rendering* volumétrico híbridos baseados em diferentes domínios são encontrados em [GLDH00,WR00].

Em [GLDH00] são descritos dois métodos de *rendering* volumétrico que utilizam *wavelets* junto com métodos baseados no domínio da imagem e no domínio dos objetos, respectivamente. O primeiro utiliza o paradigma do algoritmo de *Ray Tracing* (domínio da imagem) para representação de valores de cor e opacidade da *wavelet* cardinal *B-spline* obtendo imagens de alta qualidade. Este método utiliza elaborados esquemas de integração e avaliação dos dados para tonalização durante o *Ray Tracing* no próprio espaço *wavelet* (figura 3.3a). O segundo método utiliza um esquema rápido de integração de intensidades baseado em *splats* da *wavelet*, onde se misturam o domínio das *wavelets* e o domínio dos objetos onde se situa o algoritmo *Splatting* (figura 3.3b).



(a)



(b)

Figura 3.3. Métodos híbridos baseados em *wavelets*: (a) *Ray Tracing* baseado em *wavelets* com diferentes níveis de resolução. (b) Refinamento progressivo utilizando *Splatting* baseado em *wavelets* de Haar [GLDH00].

Outros algoritmos que utilizam a idéia das *splats* de *wavelets* também foram desenvolvidos para renderizar volumes a partir de dados em formatos comprimidos. A idéia de utilizar dados comprimidos é atenuar a necessidade de recursos de entrada/saída de dados. Isto permite que malhas tetraedrais comprimidas sejam renderizadas de forma incremental, e durante a descompressão [CTT00]. Estes algoritmos, também foram desenvolvidos para arquiteturas distribuídas como em [TCT97, CTT00, LGK02].

Em [WR00] é apresentado um método que mistura a implementação da derivada da transformada *wavelet X-ray* (domínio *wavelet*) junto com a transformada de Fourier (domínio de frequência) [RD92, VH92]. Este método aplica a transformada *wavelet* sobre o domínio da frequência. O algoritmo primeiramente calcula a transformada de Fourier tridimensional dos dados volumétricos. Para cada direção, interpola os dados transformados e reamostra-os em uma malha regular de pontos para extrair fatias utilizando um plano ortogonal à direção. Em seguida, utiliza a *wavelet* para decompor os dados dos planos no espaço de Fourier, obtendo um determinado nível de detalhe. Finalmente obtém-se a imagem aproximada mediante uma reconstrução parcial no espaço de Fourier, seguida por uma transformada inversa de Fourier bidimensional. Dado que se dispõe de coeficientes de detalhes da *wavelet* no espaço de Fourier, é possível um refinamento progressivo, embora a representação no espaço de Fourier não permita um nível de detalhe local. Para reduzir o efeito de serrilhado (*aliasing*), os dados são completados com zeros no domínio espacial (antes de aplicar a transformada de Fourier inicial em 3D).

Este tipo de técnicas, que trabalham em domínios alternativos, vem sendo explorado amplamente também para aplicações de redes e sistemas distribuídos, como veremos na próxima seção.

3.2.3. Hibridização Segundo a Arquitetura de Computação

Também encontramos algoritmos que utilizam arquiteturas mais especializadas para desenvolver técnicas tanto seqüenciais [ST90, LH91, Wi92], quanto paralelas [OUT85, JC94, EKMM91]. Considerar arquiteturas paralelas é especialmente importante no caso de simulações numéricas, que geralmente são implementadas nesse tipo de arquitetura. Os

algoritmos de visualização volumétrica paralelos são ainda mais importante no caso de arquiteturas altamente paralelas nas quais são produzidas enormes quantidades de dados volumétricos. Nestes casos é muito inconveniente mover estes conjuntos de dados para *workstations* a fim de realizar uma visualização com pós-processamento.

Além de serem consideradas as múltiplas formas de classificação das arquiteturas paralelas, deve-se levar em conta também o tipo de algoritmo de *rendering* a ser paralelizado. Uma descrição apropriada das arquiteturas paralelas e suas classificações podem ser encontradas em [AG89, Ho93]. Uma descrição das principais técnicas de medida de desempenho, como a escalabilidade e sua relação com a velocidade de processamento, pode ser encontrada em [Gu88, HM90, Wh92].

Entre as técnicas paralelas mais recentes, encontramos um exemplo de hibridização [SFLP00], que apresenta um algoritmo de *rendering* híbrido paralelo combinando duas estratégias de ordenação, *Sort-first* e *Sort-last*. Este algoritmo é dependente da posição do observador, e particiona dinamicamente em grupos, tanto o espaço bidimensional da tela, como os polígonos tridimensionais, para obter um balanceamento da carga entre os computadores de um *cluster* (figura 3.4).

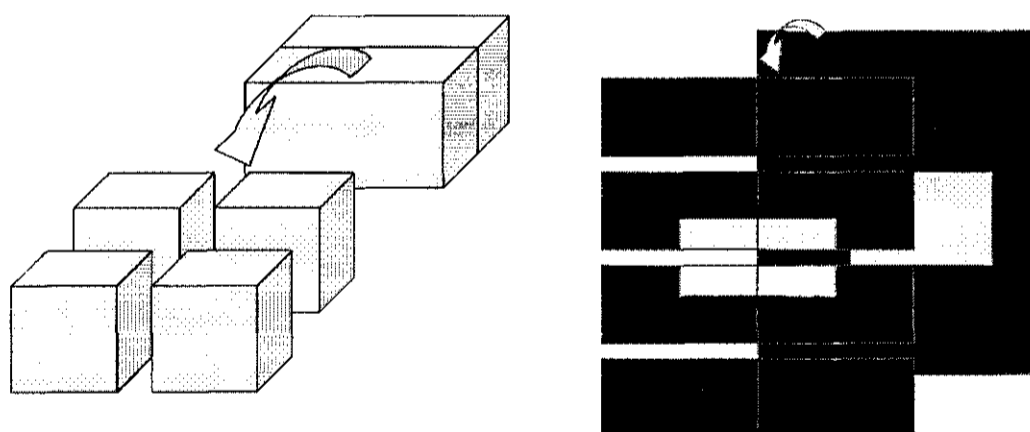


Figura 3.4. O paralelismo do espaço do objeto pode ser feito dividindo o espaço 3D do objeto em regiões separadas, e o paralelismo do espaço da imagem pode ser feito dividindo o espaço 2D da imagem em regiões.

3.3. Otimizações

Grande parte dos algoritmos de visualização podem ser vistos como otimizações a partir de uma mesma idéia básica, que foi sofrendo alterações durante sua evolução. Muitas vezes estas otimizações foram desenvolvidas paralelamente, e posteriormente agrupados. Muitos autores têm chamado esses algoritmos de híbridos, mas entretanto, são baseados em um mesmo tipo de *rendering* realizado em um mesmo espaço de projeção. Consideramos todos estes algoritmos como otimizações de uma mesma abordagem. Alguns algoritmos dessa classe incorporam técnicas dentro do *Ray Casting* ou da projeção direta mediante *Splatting*.

Um exemplo do primeiro caso é o algoritmo apresentado por Levoy e Whitaker [LW90] que está baseado no *Ray Casting*. Ele utiliza duas técnicas para reduzir o número de raios e o número de amostras para cada raio, levando em consideração à distancia do raio, a partir do ponto de visão do usuário. Imagens do volume em diferentes resoluções são armazenadas em uma pirâmide, como mostrada na *figura 3.5*. Caminhando-se pelos níveis da pirâmide é possível percorrer os diferentes níveis de resolução gerados pelo algoritmo. O nível de resolução é definido pelo numero de raios e amostras tomadas em cada raio.

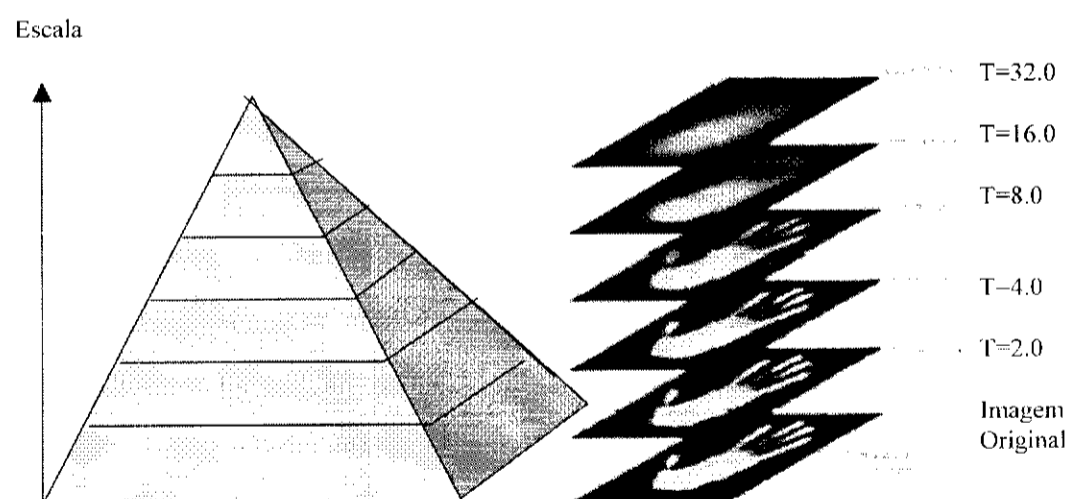


Figura 3.5. Pirâmide de Representação de Volume: O tamanho da imagem decresce quando o parâmetro de escala é incrementado.

Um outro exemplo deste primeiro caso é o algoritmo apresentado por Silva e Mitchell [SM97], que mistura diferentes técnicas de otimização no algoritmo *Ray Casting*, como a terminação adaptativa dos raios e a utilização de padrões no lançamento de raios. Este algoritmo utiliza uma aproximação do plano de varredura ou *sweep-plane*, como a proposta por Giertsen [Gi90], mas introduz diversas idéias novas que permitem uma execução mais rápida.

Um exemplo do segundo caso, projeção direta mediante *Splatting*, é o algoritmo *Hierarchical Splatting* [LH91], que apresenta um algoritmo de refinamento progressivo para o *rendering* de volumes, e utiliza uma representação piramidal do volume. A idéia do refinamento progressivo é percorrer uma malha regular sobre o plano de visualização, diminuindo a cada etapa o percurso na malha. Desta forma, a imagem é mostrada rapidamente em baixa resolução, sendo refinada até a resolução final. Como ilustra a *figura 3.6*, cada pixel da imagem é calculado uma única vez. Portanto, o tempo de cálculo da cor dos pixel permanece inalterado. A imagem é mostrada inicialmente em baixa resolução, sendo refinada até atingir a resolução da imagem final. Esta retroalimentação imediata permite que o usuário interrompa o processo de visualização para, por exemplo, escolher outro ponto de visão.

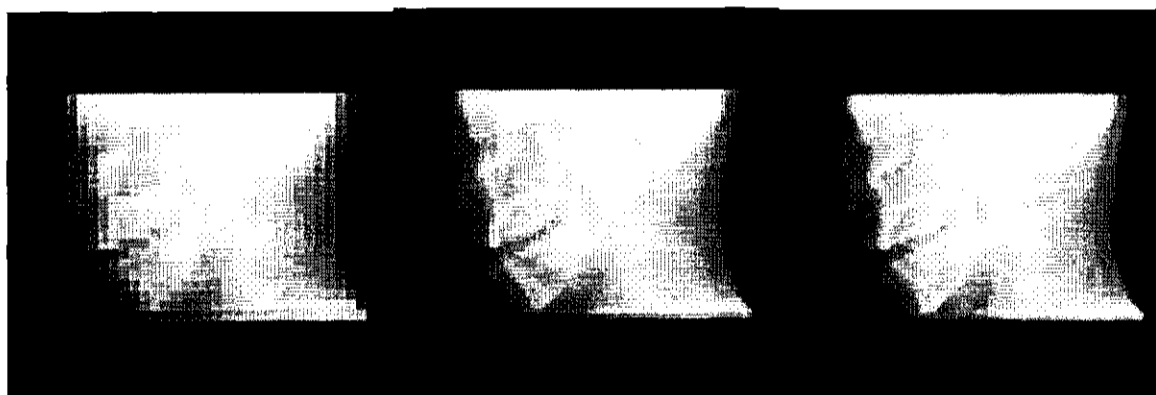


Figura 3.6. Refinamento Progressivo de Hugues Hopes.

Utilizando uma *octree* encaixada na pirâmide, o algoritmo possibilita que o usuário defina a precisão na visualização. A *octree* é projetada usando um conjunto de *splats*, ou *footprints*, cada uma escalonada para se adaptar ao tamanho da célula de projeção. Os *splats* são aproximados para polígonos, utilizando o modelo de iluminação de Gouraud em *RGBA*, e

assim são ser desenhados eficientemente em *workstations* modernos. O refinamento progressivo é utilizado para evitar o desperdício de memória e obter uma maior interatividade com o usuário. Esta idéia do refinamento progressivo é explorada também por outros algoritmos de otimização [Ka91,KSS00] e híbridos [KYC90].

Para aproveitar a coerência espacial no desenvolvimento de algoritmos eficientes, alguns pesquisadores têm também combinado técnicas de projeção com a produção de primitivas geométricas renderizáveis por *hardware*. Tais técnicas aumentam a velocidade do processo de *rendering* volumétrico direto, mas muitas vezes provocam um decréscimo na qualidade e exatidão da imagem.

Alguns exemplos são os algoritmos apresentados em [ST90, Wi92, LH91]. No trabalho de Shirley e Tuchman [ST90] consegue-se uma projeção eficiente de células tetraedrais, representando as células como triângulos semitransparentes renderizados por *hardware*. Laur e Hanrahan [LH91] combinam uma representação piramidal com geração de primitivas renderizáveis por *hardware*. As primitivas renderizáveis são escalonadas dependendo do nível de resolução utilizada pelo usuário. Williams [Wi92] ainda explorou várias aproximações do algoritmo de Shirley e Tuchman [ST90] para projetar células tetraedrais.

Arquiteturas especializadas têm sido desenvolvidas para acelerar o *rendering* de aplicações, tais como o *rendering* a partir de imagens médicas e modelagem de sólidos (*figura 3.7*). Uma análise de alguns destes sistemas é por Kaufman [KBCY90]. Muitas destas arquiteturas são desenhadas inicialmente para suportar *voxels* opacos e não modelos volumétricos semitransparentes, ainda que algumas consigam isto sacrificando o desempenho.

As arquiteturas especializadas propostas e desenvolvidas para aplicações de modelagem de sólidos incluem: o 3DP4 [OUT85], Insight [Me82], PARCUM II [JC94], e a Ray Casting Engine [EKMM91]. Outras arquiteturas, desenvolvidas para aplicações médicas, são: o Voxel Processor [GRBB85], e o Cube [KB88]. Todas estas arquiteturas são do tipo MIMD (*Multiple Instruction and Multiple Data* – seção 4.5.6), tendo memória compartilhada ou distribuída [Si95]. Duas delas são do tipo *pipelined* [OUT85, EKMM91] com diferentes processadores encarregados de diferentes tarefas no *pipeline* de *rendering*. Nestes casos, as conexões entre memória e processos são específicas das tarefas que estão sendo executadas. A memória poderia também ser especializada para acelerar o processo de *rendering* [KB88], permitindo leituras e escritas simultâneas.

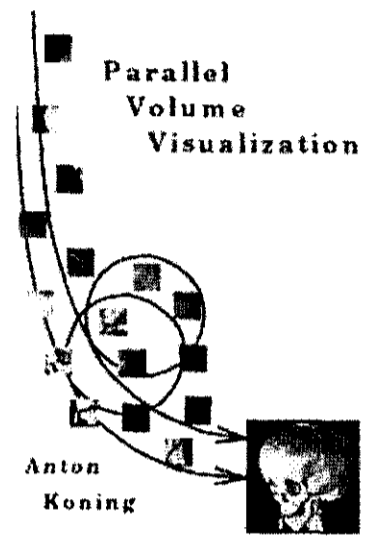


Figura 3.7. Rendering volumétrico paralelo, por Anton Koning.

Challinger [Ch92] apresenta um algoritmo de *rendering* paralelo de volumes para malhas computacionais não retilíneas. Este algoritmo é generalizado para trabalhar com malhas e células não convexas, malhas com vazios, malhas construídas de multiblocos, e primitivas geométricas embutidas. Este algoritmo também foi projetado para uma arquitetura do tipo MIMD, com características de memória local ou compartilhada, com tempos de acesso à memória não uniformes. Porém, apresenta alguns problemas com o balanceamento de carga e uma certa perda de coerência, devido às tarefas de decomposição do espaço da imagem.

Outra técnica que considera arquiteturas distribuídas foi apresentada por Lippert et al. [LGK02]. Ela trabalha com dados volumétricos comprimidos que são renderizados utilizando um algoritmo híbrido de *Splatting* baseados em *wavelets*. Este método foi especialmente projetado para aplicações distribuídas e de redes, assumindo um servidor remoto para manter grandes conjuntos de dados volumétricos que são revisados, listados e renderizados interativamente por um cliente local. É por isso que os dados volumétricos do servidor são codificados utilizando um método de compressão de volumes baseado em *wavelets*. Um cliente local pode renderizar um volume, imediatamente, desde o domínio de compressão, utilizando *wavelet footprints* (similares às *footprints* do algoritmo *Splatting*). Assim, as imagens são

refinadas progressivamente, à medida que os dados vão chegando. A qualidade da imagem dependerá muito da largura de banda ou da capacidade de comunicação do cliente.

Como estes métodos, muitos outros têm sido desenvolvidos, visando sempre obter uma maior interatividade com o usuário. Conseguir aplicações em tempo real que suportem as grandes quantidades de dados volumétricos produzidos por diferentes aplicações, apresenta-se ainda como um desafio, o que faz da visualização volumétrica uma área de pesquisa muito ampla e rica.

3.4. Considerações Finais

Neste capítulo, após uma breve introdução, foi apresentada uma visão geral das principais tendências na hibridização de algoritmos de *rendering* utilizadas pelas novas técnicas de Visualização Volumétrica.

Para exemplificar melhor cada uma destas tendências, foram descritos alguns dos algoritmos mais salientes de cada tendência, seguindo as diferentes abordagens de *rendering* volumétrico e seus métodos mais representativos.

Os poucos exemplos apresentados neste capítulo proporcionam uma idéia da grande amplitude de possibilidades que oferece os algoritmos híbridos, obtendo-se vantagens variadas, desde o melhor aproveitamento de recursos até uma melhor cobertura das necessidades dos usuários. Tudo isto sempre dependendo do tipo de aplicação que se deseje desenvolver.

CAPÍTULO 4

Visualização Volumétrica Baseada no Ray Casting

4.1. Considerações Iniciais

Devido ao alto tempo de processamento da visualização volumétrica direta, comparado com a visualização por iso-superfícies, é de suma importância identificar formas de otimizar e simplificar os algoritmos de visualização disponíveis na literatura, de forma a possibilitar a sua utilização em diversas áreas de aplicação.

Muitos dos algoritmos de visualização são baseados no algoritmo *Ray Casting*, desenvolvidos tanto para visualizar dados de malhas regulares [Le90a, Le90d, Fr91, MK92, YCK92, SGDM95, Si95, Ph97, Sc99a], quanto de malhas irregulares [Ga90, RW92, MMS94, Fr94, BKS97, SM97]. O *Ray Casting* permite a visualização de imagens de alta qualidade e possibilita de forma natural, a visualização de estruturas internas através do uso de transparência nas estruturas mais externas.

Basicamente, os algoritmos de *Ray Casting* calculam a interseção dos raios com o volume e polígonos em separado e, a seguir, combinam as interseções ao longo do raio. O algoritmo percorre os pixels do plano da imagem, a partir dos quais se disparam os raios, encontrando uma cor e opacidade para cada um. Estas opacidades e cores de iluminação encontradas ao longo do raio se sintetizam para calcular a opacidade e a cor final do pixel. Um algoritmo similar ao *Ray Casting* é o *Ray Tracing*. Neste algoritmo, os raios rebatem quando alcançam objetos reflexivos, enquanto no *Ray Casting*, os raios continuam em linha reta até que a opacidade encontrada pelo

raio seja igual a um limiar determinado, por exemplo 1, ou o raio saia pela parte posterior do volume.

Tipicamente, a complexidade do algoritmo de *Ray Casting*, assim como dos algoritmos de projeção, é $O(n^3 + s^2 n)$, onde n é a quantidade de valores em cada dimensão do volume (assim, o volume tem n^3 amostras), e s é o número de pixels em cada dimensão da imagem (uma imagem s^2 está sendo criada), conforme mostrou Challinger [Ch93] para o algoritmo de *Ray Casting*.

Nos últimos anos, vários pesquisadores apresentaram otimizações e estratégias alternativas e híbridas com o objetivo de aumentar a velocidade dos algoritmos. Geralmente, estes métodos consideram uma representação mediante malhas cartesianas de células, mas os novos métodos de visualização preferem adotar um tipo mais geral de malha (malhas irregulares não estruturadas), como já foi mencionado. O desenvolvimento de algoritmos que utilizem este tipo de malha não estruturada ainda é incipiente e tem que superar diversos problemas. Em geral, o *rendering* volumétrico de malhas irregulares é uma operação difícil e existem muitas abordagens diferentes para este problema. O mais simples e ineficiente é amostrar novamente a malha irregular para uma malha regular. Geralmente, para obter a precisão requerida, um valor de amostragem suficientemente alto tem que ser usado, o que, na maioria dos casos, fará o volume da malha regular resultante demasiadamente grande para propósitos de armazenagem e *rendering*, sem mencionar o tempo para executar novamente a amostragem.

A fim de entender melhor os problemas apontados acima, faremos uma revisão detalhada do algoritmo *Ray Casting* e de suas variantes.

Este capítulo apresenta, na *seção 4.2*, os trabalhos correlatos baseados no *Ray Casting*. Na *seção 4.3*, é definido o algoritmo de *Ray Casting* Binário. Na *seção 4.4*, apresenta-se o algoritmo de *Ray Casting* desenvolvido por Levoy. A *seção 4.5* foi dedicada à apresentação das diferentes otimizações do *Ray Casting* em suas diferentes etapas: lançamento dos raios, determinação dos segmentos dos raios que interceptam o volume, exploração da coerência espacial do volume, acumulação das contribuições de opacidade e cor ao longo do raio, exibição do plano de visualização durante sua construção e, considerando-se também como uma otimização, a aceleração utilizando algoritmos paralelos. Finalmente, na *seção 4.6* estão contidas as considerações finais deste capítulo.

4.2. Trabalhos Correlatos

Um numeroso grupo de pesquisadores trabalharam sobre a idéia da técnica de *Ray Casting* apresentada por Levoy [Le88]. Sabella [Sa88] modificou o algoritmo original de Levoy, considerando cada célula como um emissor de densidade variável. Neste método, quatro valores são calculados para cada raio lançado através do volume: o máximo valor encontrado ao longo do raio, a distância desde a origem do raio até este valor máximo, a intensidade da atenuação, e o centro de gravidade. A imagem é gerada usando o modelo de cor HSV (*Hue*, *Saturation* e *Value*) da seguinte forma: o máximo valor encontrado é a cor (*hue*), a distância para o valor máximo ou para o centro de gravidade é a medida da saturação de cores (*saturation*), e a intensidade da atenuação é o valor (*value*). No algoritmo original, Levoy utiliza os próprios valores nas células para determinar cor e opacidade.

Posteriormente, começaram a surgir algoritmos que otimizaram o algoritmo de *Ray Casting* objetivando reduzir o consumo de memória e o alto custo computacional. As primeiras otimizações foram propostas pelo próprio Levoy [Le90c]: 1) substituindo a enumeração exaustiva do volume por uma enumeração espacial hierárquica, de forma que as células de valores semelhantes são agrupadas em blocos de células, e, 2) utilizando a terminação do raio de uma forma adaptativa, interrompendo o cálculo das contribuições quando estas não forem mais significativas.

Lacroute e Levoy [LL94] apresentaram um algoritmo que combina as vantagens de outros algoritmos, fazendo uso da coerência no volume e na imagem, obtendo um acesso sincronizado às estruturas de dados, mantendo sempre o alinhamento entre o “caminhamento” no volume e no plano de visualização.

Kaufman et al. [KYC90] desenvolveram um algoritmo híbrido que utiliza dois *Z-buffers*, um para armazenar superfícies e o outro para armazenar o volume de dados, que são combinados no final para gerar a imagem da cena completa. Entretanto, como os *Z-buffers* foram calculados de modo independente, este algoritmo não permite tratar volumes semitransparentes.

Zuffo e Lopes [ZL94] apresentaram uma otimização significativa das etapas de visualização (*pipeline*) originais de Levoy: antecipar a reamostragem, fazendo-a sobre os dados originais, e eliminando o volume intermediário com as componentes RGBO (cor e opacidade). Além disso, as imagens finais têm uma melhor qualidade devido à melhor aproximação do

gradiente gerada a partir dos dados reamostrados, e não a partir dos dados originais como propôs Levoy.

Yagel e Kaufman [YK92] desenvolveram uma técnica de visualização volumétrica baseada em modelos (*Templates*), que utiliza um padrão de discretização que é utilizado para todos os raios lançados no volume. Para garantir que cada célula contribua apenas uma vez na imagem, e evitar que outras não sejam consideradas, é necessário que o raio seja do tipo *26-connected*, ou seja, caso seja retirado qualquer ponto, o modelo (*template*) perde a continuidade. Além disso, é necessário que os raios sejam lançados a partir do volume de dados e não do plano de visualização.

Uma outra otimização faz o lançamento dos raios através da determinação de pontos de referência no espaço do objeto. O caminhar na imagem é efetuado através de incrementos previamente calculados neste plano de referência. Estas otimizações simplificam a proposta de Yagel e Kaufman, pois se utiliza dois planos auxiliares de visualização.

Silva e Kaufman [SK95] apresentam uma técnica híbrida de *Ray Casting* assistido por polígonos. Esta técnica de otimização consiste basicamente em reduzir ao máximo os limites do volume para logo discretizar cada raio lançado no volume. Para isto, é necessário desprezar as células transparentes no começo e no fim do percurso de cada raio. O algoritmo, primeiramente, encontra uma representação poliedral que contenha todas as células não transparentes. Fora desta representação, as demais células têm opacidades inferiores a um valor de limiar especificado pelo usuário. A implementação desta técnica envolve a projeção da representação poliedral que contém o volume em dois *Z-buffers*: o primeiro, convencional, guarda a menor distância do observador ao volume, enquanto o outro guarda a maior distância. Com isso, tem-se o limite inferior e superior para a discretização de cada raio lançado desde os pixels do plano da imagem. Por ser uma técnica paralela, deve-se decidir a forma de designar a carga de trabalho de cada processador. O principal problema na implementação paralela desta aproximação é que, para o *Ray Casting* ser eficiente, a parte do conjunto de dados de cada processador deve ser contíguo no espaço. Isto torna a composição mais fácil e também reduz o número de cálculos de interseção requeridos.

Garrity [Ga90] desenvolveu um algoritmo de *Ray Tracing* para dados volumétricos irregulares, ele subdivide as células em tetraedros, se não forem já desse tipo, isto permite fazer uma interpolação direta quando se utilizam coordenadas homogêneas. Outra das diferenças deste

método é que utiliza um grafo de adjacência das células explícito. Esse grafo ajuda seguir o raio através da malha não estruturada, para então determinar o ponto amostrado da próxima célula.

Usselton [Us91], Ramamoorthy e Wilhelms [RW92], e Challinger [Ch92] apresentaram algoritmos de *Ray Tracing* desenvolvidos para malhas curvilíneas. Estes autores dedicaram-se resolver o problema da interpolação dos dados para a obtenção das amostras nestas malhas curvilíneas. As amostras são tomadas unicamente nas interseções entre os raios e as faces das células, isto faz com que a distancia entre as amostras não seja regular, o que deve ser tido em conta durante o cálculo das amostras de cor e opacidade. Ramamoorthy e Wilhelms descreveram detalhadamente como calcular as interseções entre o raio e as células e interpolar dentro da face da célula, que é um quadrilátero irregular. Challinger primeiro triangula todas as faces da célula e armazena-as em listas de triângulos. Este método toma vantagem das arquiteturas paralelas para realizar o *rendering* das primitivas geométricas.

Fruhauf [Fr94] desenvolveu uma técnica de *Ray Casting* para dados volumétricos curvilíneos. Os raios não são enviados através do “espaço físico” que é irregular se não através do espaço computacional que é regular. É uma espécie de reamostragem para uma malha regular mantendo o tamanho, qualidade e topologia da informação. Este método não pode ser aplicado a dados não estruturados.

Bunik e Kaufman [BKS97] desenvolveram um algoritmo de *Ray Casting* para malhas não estruturadas tetraedrais sendo possível estende-lo para outras malhas. A idéia básica é renderizar as faces das células e não as células como um todo, mantendo cada vértice uma lista dos triângulos que o compartilham. Após é seguido o algoritmo de *Ray Casting* para o *rendering*. Ao igual que Garrity [Ga90] que utiliza tetraedros, a ordem da profundidade está baseada na ordem implícita providenciada pela informação de conectividade das células.

Silva e Mitchell [SM97] desenvolveram um algoritmo híbrido de *Ray Casting* utilizando *sweep planes* para o *rendering* de malhas irregulares gerais. Este algoritmo está baseado no paradigma do *sweep-plane* para acelerar o *Ray Casting* para malhas irregulares não estruturadas, não convexas, e desconexas (com buracos), sendo que o custo do algoritmo decresce com a desconectividade. Este algoritmo explora a coerência espacial sem importar se a resolução da imagem difere muito da resolução do espaço do objeto.

Mesmo com as técnicas de otimizações já existentes, a maioria dos algoritmos seqüenciais não consegue a visualização volumétrica em tempo real. O tempo mínimo obtido é

chamado de “tempo para viabilizar interação” (*interactive-time*), indicando que os computadores atuais ainda não são rápidos o suficiente para gerar imagens de alta qualidade, juntamente a interação [SFLP00].

4.3. O Algoritmo de *Ray Casting* Binário

Uma das primeiras técnicas de *rendering* de volumes do tipo *image-order* foi desenvolvida por Tuy e Tuy [TT84], esta técnica foi nomeada como *Ray Casting* Binário (*Binary Ray Casting*). Esta técnica gera imagens de superfícies contidas dentro de dados volumétricos binários, sem necessidade de uma detecção explícita da fronteira ou bordo dos objetos, nem da remoção de superfícies escondidas. Para prover todas as vistas possíveis, os dados são mantidos em uma posição fixa e o plano da imagem é movimentado.

Este algoritmo envia um raio a partir de um pixel e determina sua interseção com a superfície do objeto. Para projeções paralelas, todos os raios são paralelos à direção de visão. Para projeções perspectivas, os raios são lançados a partir do ponto do observador, de acordo com a direção e campo de visão. Um exemplo bidimensional de cada projeção, paralela e perspectiva, são mostrados na *figura 4.1*. O sombreamento é feito quando acontecer uma interseção do raio e o objeto, obtendo-se a cor resultante do pixel.

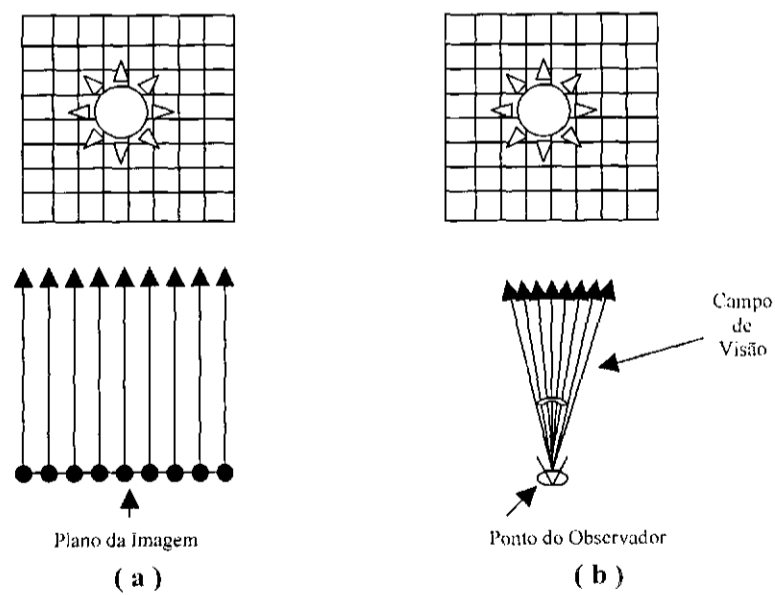


Figura 4.1. Projeção: a) Projeção paralela. b) Projeção perspectiva.

Para determinar a primeira interseção ao longo do raio, é utilizada uma técnica por etapas, de forma que o valor da amostra é determinado em intervalos regulares ao longo do raio, até atingir o objeto. Amostras com valor 0 são consideradas fora do objeto ou parte do fundo (*background*), e aquelas com um valor 1 são consideradas parte do objeto. Uma técnica de interpolação de ordem zero é usada para calcular qual é a posição que será amostrada ao longo do raio. Se um raio apresenta somente pontos de amostra com um valor 0, o raio perdeu o objeto por completo. Se existe, ao longo do raio, alguma amostra p_i , tal que todas as amostras p_j , para $j < i$, têm um valor 0, e a amostra p_i tem um valor 1, então, o ponto de amostra p_i é considerado como a primeira interseção ao longo do raio. Um exemplo 2D desta técnica por etapas é mostrado na *figura 4.2*, onde o valor de p_0 até p_3 é 0, e o valor de p_4 é 1.

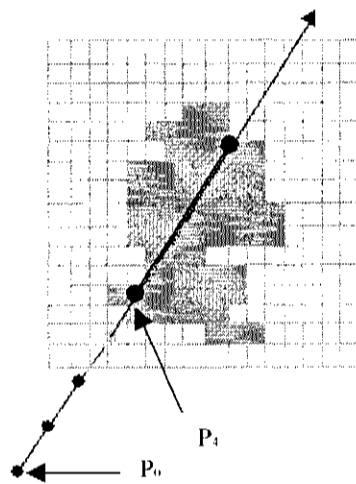


Figura 4.2. Técnica bidimensional por etapas.

4.4. O Algoritmo de *Ray Casting*

O algoritmo de *Ray Casting* foi originalmente proposto como uma técnica que permitia a visualização de pequenos detalhes internos do volume, por meio do controle da transparência das células.

O algoritmo de *Ray Casting* permite a geração de imagens de alta qualidade através do *rendering* direto, operando no espaço da imagem. O algoritmo clássico de *Ray Casting* [Le90d]

baseia-se em raios lançados a partir do ponto de visão do observador, que passam através de cada pixel da tela (plano de visualização ou plano de projeção) e intersectam o volume. Para cada raio lançado, uma amostragem é feita com base nos valores dos *voxels* (ou células) e a cor final de cada pixel da imagem é obtida integrando as contribuições de cor e opacidade de cada *voxel* interceptado pelo raio. Quando a projeção é paralela, a direção destes raios é a da normal ao plano de projeção (figura 4.3).

Existem basicamente duas abordagens para amostrar o raio. A primeira envolve a localização do ponto de entrada do raio no volume e, por meio de divisões eqüidistantes, a amostragem é feita até se atingir o ponto de saída do raio no volume. Dependendo do tamanho da divisão, é possível que não seja incluída a contribuição de alguns *voxels*. A segunda abordagem envolve a determinação dos pontos de entrada e saída para cada um dos *voxels* interceptados pelo raio. A contribuição de cada *voxel* é, então, calculada por interpolação.

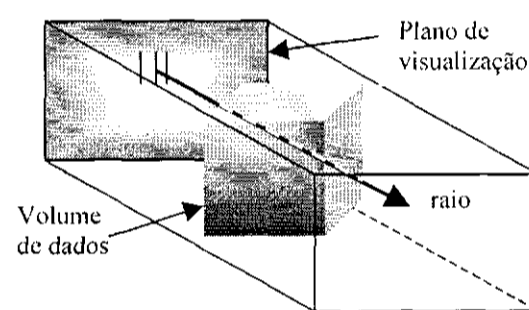


Figura 4.3. Algoritmo de Ray Casting

A abordagem original de Levoy [Le90a] executa uma série de etapas, listadas a seguir.

No primeiro passo do algoritmo de *Ray Casting*, o usuário estabelece tabelas de classificação dos dados. O usuário também deve especificar o ponto de visão, e os parâmetros da iluminação e de visualização da cena. Esta primeira etapa inclui a preparação de dados. Os valores adquiridos na localização do *voxel* (ainda não processados) são colocados em um *array*. Esta preparação de dados pode incluir a correção dos dados e o melhoramento mediante interpolação de amostras adicionais.

Para o cálculo do *buffer* de profundidade, cada ponto no plano da imagem define um raio através dele, iniciando-o no ponto de visão do observador. Então, um raio é enviado a partir de

um pixel no plano da imagem. Ao longo do raio, amostram-se vários segmentos até atingir o outro extremo do volume ou um número máximo de amostras. Esta etapa do algoritmo pode ser paralelizada com um número arbitrário de *threads* calculando as linhas de varredura (*scanlines*) em paralelo.

Em seguida, utiliza-se uma função de transferência para determinar a cor e a opacidade que o usuário especificou para os valores dos dados. O *array* de valores resultante da preparação dos dados, agora é usado como entrada para o modelo de iluminação nas interseções, obtendo a cor final de cada pixel. O modelo de tonalização associa uma cor a cada posição, seguindo geralmente o modelo desenvolvido por Phong [PT75].

Após isto, é feita a reamostragem do volume de dados óptico (volume *RGBO*), seguindo o paradigma do *Ray Casting*. Cada pixel da imagem final é calculado acumulando a contribuição da opacidade e cor de cada ponto reamostrado ao longo do raio correspondente.

Métodos de interpolação são utilizados para determinar cada ponto de reamostragem, usando informação da vizinhança. Tanto o *Ray Casting* binário como este *Ray Casting* discreto utilizam interpolação de ordem zero para definir o valor escalar em qualquer ponto de R^3 . Uma vantagem desta interpolação é a simplicidade e velocidade, dado ao fato que muitos dos cálculos processados podem ser feitos usando aritmética com números inteiros. Uma desvantagem é o efeito de serrilhado (*aliasing*) na imagem, devido à limitada precisão numérica usada ao calcular a composição *RGBO*, e à interpolação linear das opacidades [SBM94].

Interpolações de ordem superior (triquadráticas e tricúbicas) podem ser usadas para criar uma imagem mais exata, mas geralmente incrementam o custo do tempo de cálculo e a complexidade do algoritmo.

Levoy implementou este algoritmo através de dois *pipelines* independentes: um para iluminação e um para classificação do material, resultando em uma fase final de composição de ambos *pipelines* (figura 4.4) para a geração da imagem final utilizando simples interpolações lineares.

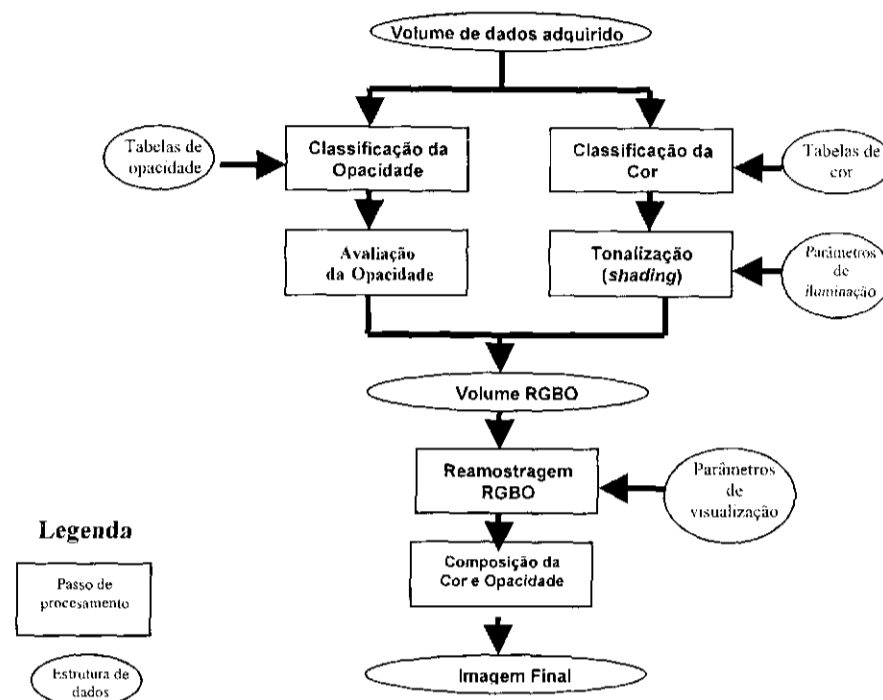


Figura 4.4. Pipeline de rendering direto de Levoy [ZL94].

4.4.1. Classificação

A classificação dos dados é a operação mais difícil que o usuário de um sistema de visualização tem que realizar [KB97]. A finalidade deste processo é identificar as estruturas internas do volume. Por exemplo, um médico pode isolar elementos distintos, tais como gordura, músculo e osso, em um exame de ressonância magnética [CLLC88]. No caso dos algoritmos de *rendering* direto, cujos dados são escalares, a classificação dos dados significa eleger valores de cor (brilho) e opacidade (atenuação da luz) a cada conjunto de valores, mediante a definição de funções de transferência. Isto nos mostra que a classificação é altamente dependente do tipo de dado a ser visualizado.

Existem algumas técnicas automáticas de classificação do volume [DCH88], mas na maioria dos casos ainda é necessária a interferência do usuário para definir adequadamente as funções de transferência, visto que para obter uma imagem razoável, o usuário deve escolher as tabelas de classificação com cuidado. Felizmente existem ferramentas capazes de auxiliar o usuário no processo de classificação, como o histograma do volume. Por exemplo, os valores das tabelas utilizadas para visualizar dados de uma tomografia computadorizada deveriam projetar e

apresentar valores de densidade de ossos (branco ou opaco), valores de densidade de músculos (vermelho ou semitransparente), e valores de densidade de tecido gorduroso (creme ou muito transparente). Pequenas variações nos valores da densidade e opacidade, freqüentemente têm um grande e inesperado impacto na imagem visualizada.

Tipicamente, calcula-se um vetor de cores e opacidades para cada raio, reamostrando o banco de dados da célula. Um fundo totalmente opaco é disposto atrás do conjunto de dados, e as cores e opacidades reamostradas são fundidas mediante composição, desde o plano posterior até o plano frontal (tipo de ordenamento “*back-to-front*”) a fim de obter uma única cor para o raio.

A função de transferência de opacidade permite mapear valores contidos em um volume a uma determinada transparência. Esta transparência normalmente é representada por um número real no intervalo de 0.0 (para totalmente transparente) a 1.0 (para totalmente opaco) [CV00]. Assim, uma opacidade $o(x)$ baseada na densidade dos materiais, é associada a cada célula. A composição final $o'(x)$ pode ser realizada enfatizando as bordas das regiões de densidade quase uniforme, e atenuando o interior, multiplicando o valor da opacidade $o(x)$ pelo gradiente $\nabla f(x)$:

$$o'(x) = o(x) |\nabla f(x)|.$$

A visualização final da imagem é feita pela projeção bidimensional da cor $c(x)$ e opacidade $o(x)$ da célula x , no plano de visualização.

Outros fatores que devem ser considerados na classificação são: o conhecimento do usuário sobre o material que está sendo classificado, a posição dos elementos classificados no volume, e o espaço que ocupa o material do elemento [DCH88]. Se o usuário está familiarizado com o material que está classificando, a especificação das tabelas é questão de escrever alguns números. Mas, se o usuário não está familiarizado com o material, pode-se requerer uma exploração intensiva dos dados ou ajuda de um especialista. Geralmente, os algoritmos não consideram o espaço parcial ocupado pelo material, e não implementam a capacidade de múltiplas seções de materiais diferentes.

4.4.2. Iluminação

Um modelo de iluminação é utilizado para criar a ilusão de profundidade, realçar bordas e características do volume na visualização. Tipicamente, os algoritmos de visualização utilizam aproximações de modelos de iluminação de baixo espalhamento, pois computacionalmente são

mais simples e apresentam bons resultados, especialmente para imagens médicas. A utilização de modelos de alto espalhamento proporcionaria maior realismo, porém, em casos em que se utiliza transparência para a análise de estruturas internas não traz grandes benefícios para sua interpretação. Portanto, na maioria dos casos, não se justifica sua utilização.

Os modelos de baixo espalhamento utilizam um modelo local de iluminação, que consideram apenas a reflexão da luz na superfície do objeto. Nestes modelos, apenas a luz ambiente e as reflexões, difusa e especular, das fontes de luz existentes na cena, são consideradas no cálculo da iluminação. Desta forma, as componentes *RGB* da cor $c(x)$, para cada célula x , são calculadas a partir de uma estimativa do gradiente da função densidade $d(x)$ e da intensidade de luz, usando algum modelo de iluminação, como por exemplo, Gouraud [Go71] ou Phong [PT75].

Estes modelos consideram a utilização de uma ou múltiplas fontes de luz. Tipicamente, a luz emitida pelas fontes de luz presentes na cena não é atenuada pelo volume. Este efeito é usualmente ignorado durante o cálculo da iluminação, pois além de simplificar bastante o algoritmo (não é necessário calcular uma integral), muitas vezes é até indesejado. Se as fontes de luz fossem atenuadas pelo volume, áreas ao redor de regiões muito densas (por exemplo, osso) poderiam ficar totalmente escuras. Entretanto, note que o efeito da atenuação da luz ao longo do raio de visão não pode ser desprezado. Isso é levado em consideração durante a etapa de projeção do *pipeline* de visualização.

No caso de algoritmos de *rendering* direto, o conceito de superfície pode não fazer muito sentido, mas é importante para o correto cálculo da iluminação das células. Efetivamente, os algoritmos procuram fazer uma estimativa da normal da superfície para efeito de iluminação.

4.4.3. Projeção

Esta etapa realiza a projeção das células sobre o plano da imagem. Tipicamente, a projeção é realizada efetuando o lançamento de um raio a partir de cada pixel da tela (plano de projeção), determinando, por meio de composição da cor iluminada e opacidade das células atingidas pelo raio, a cor final do pixel.

A projeção pode ser paralela ou em perspectiva. Entretanto, a projeção em perspectiva possui algumas desvantagens. Uma das maiores é o problema de divergência dos raios. A

densidade de raios – o número de raios lançados por célula – diminui à medida que se caminha sobre o raio. Ou seja, a amostragem realizada nas células mais próximas do observador é mais detalhada que a realizada nas células mais distantes. Isso significa que pequenos detalhes são perdidos ao serem projetadas as células mais distantes. Para resolver este problema, pode-se usar um algoritmo adaptativo, aumentando-se a densidade de raios à medida que se afasta do observador [PSG99].

O algoritmo *Ray Casting* gera imagens de alta qualidade. Entretanto, seu custo computacional pode ser muito grande, dependendo da resolução da imagem final (da quantidade de raios a serem lançados através do volume). Uma outra desvantagem, é que os raios lançados geralmente atravessam o volume fora da ordem de armazenamento das células, sendo necessário calcular em qual célula está cada ponto amostrado ao longo do raio.

4.5. Otimizações Sobre o Algoritmo Original

Como já foi mencionado, o algoritmo original descrito por Levoy consome muita memória e tem alto custo computacional. Por isto, o próprio Levoy propôs duas otimizações.

A primeira delas substitui a enumeração exaustiva do volume por uma enumeração hierárquica espacial, em que *voxels* de valores semelhantes são agrupados em células. Alguns métodos foram propostos para reduzir o trabalho computacional, explorando a coerência espacial entre os *voxels*. Podem ser utilizadas técnicas como “dividir para conquistar” (“*divide to conquer*”), tabelas pré-calculadas, ou transformações incrementais. Um exemplo da exploração da coerência do espaço das células é utilizar uma representação do volume 3D mediante uma *Octree*, ou mediante as árvores *BSP* que serão descritas neste capítulo.

A segunda otimização refere-se à terminação do raio de uma forma adaptativa, na qual interrompe-se o cálculo das contribuições quando estas não forem mais significativas.

Note-se que, tendo que armazenar densidade, cor (R, G, B) e opacidade para cada célula, um volume de $256 \times 256 \times 256$ requer quase 90 Mbytes de memória. Para contornar este problema, os cálculos podem ser feitos no tempo do lançamento dos raios, diretamente sobre o valor da densidade, resultando em uma redução para 16.7 Mbytes. Para que este cálculo não fique muito lento, pode-se adotar um modelo simples, que associa, a cada valor de densidade, uma cor e uma transparência, utilizando-se uma tabela.

Os métodos existentes seguem um ou mais dos seguintes princípios para acelerar o processo de *Ray Casting*:

- *Coerência do espaço dos pixels*. Dado que é altamente provável que entre dois pixels de cor parecida (ou idêntica) exista um outro pixel com iguais propriedades, diminui-se a necessidade de enviar um raio para atravessar tal pixel.

- *Coerência do espaço dos objetos*. Seguindo o princípio anterior aplicado ao conceito 3D do espaço com células, evita-se amostrar as regiões do espaço 3D que apresentam valores similares.

- *Percursos do espaço*. Considerando que um raio atravessa o espaço em duas etapas – 1) o raio avança em um espaço vazio até achar um objeto, para logo 2) integrar as cores e opacidades na vez que o raio atravessa o objeto –, pode-se acelerar o processo omitindo a primeira etapa.

Desta maneira, algumas etapas do *Ray Casting* que podem ser otimizadas são:

- Lançamento dos raios;
- Determinação dos segmentos dos raios que interceptam o volume;
- Exploração da coerência espacial do volume;
- Acumulação das contribuições de opacidade e cor ao longo do raio;
- Exibição do plano de visualização durante a “construção” da imagem.

A seguir, definiremos como pode otimizar-se cada uma destas etapas.

4.5.1. Lançamento dos Raios

Uma abordagem para otimizar o lançamento dos raios no algoritmo *Ray Casting*, é aproveitar a noção de coerência entre os raios, apresentada na *figura 4.5*. Um método que explora esta idéia é o Método Baseado em Modelos (*Template-based*) [YK92]. Como na projeção paralela existe uma grande coerência entre os raios (os raios têm a mesma inclinação ou “forma”, ainda que tenham diferentes origens), o conjunto de passos que estes raios seguem quando

atravessam um volume é similar. Aproveita-se esta coerência ao evitar o cálculo que significa direcionar o raio através do espaço da célula.

Assim, neste método precisa-se calcular uma vez só a forma do raio e armazená-la em uma estrutura de dados padrão (nomeada “*ray-template*”). Desta forma, todos os raios podem ser gerados seguindo este padrão. Deve-se ter cuidado com a posição de cada raio.

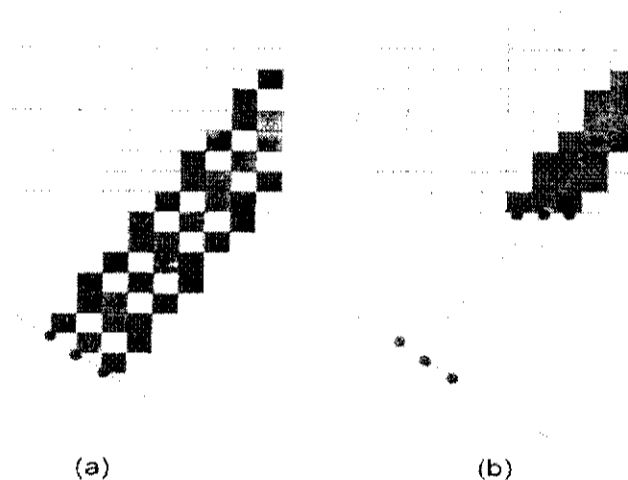


Figura 4.5. Ray Casting discreto: a) do plano da imagem b) e plano da base.
Base do algoritmo “*Template-based volume viewing*” [YK92].

Este algoritmo começa encontrando o plano base, aquele onde se projeta a maior área do volume, e o modelo padrão (“*ray-template*”). O plano base é um dos três planos (XY, XZ, ou YZ) do espaço do objeto, no qual o volume projeta a área maior.

A partir deste plano e da região projetada do volume, raios paralelos são testados no volume, repetindo a seqüência de etapas especificadas no modelo padrão. Como resultado, obtemos a projeção do volume no plano base. A terceira fase do algoritmo transforma a imagem projetada desde o plano base ao plano da tela.

Mais recentemente, algumas implementações utilizando computadores paralelos de alto desempenho têm sido apresentadas. No entanto, devido ao alto custo dos computadores paralelos e à disponibilidade de computadores de uso geral conectados em redes locais, o uso de

computação distribuída tem sido explorado. Esta abordagem apresenta uma melhor relação custo/benefício do que com computadores massivos paralelos [Ma95].

4.5.2. Determinação dos Segmentos dos Raios que Interceptam o Volume

Para calcular a cor final de um pixel, utilizando o algoritmo *Ray Casting*, é necessário discretizar cada raio lançado e acumular a contribuição de cor e opacidade das células ao longo do raio. Essa operação consome bastante tempo de computação, pois precisa ser realizada em todos os raios lançados no volume.

Especificamente, na determinação do tamanho dos segmentos de amostragem, o tamanho d deve ser cuidadosamente escolhido. Se d é muito grande, pequenas características nos dados poderiam não ser detectadas. Mas, se d é muito pequeno, o custo computacional é muito alto.

Algumas otimizações dizem respeito à detecção da interseção do raio com o volume. Para tal, são utilizadas extensões do algoritmo clássico de Cyrus-Beck para *clipping*.

Para a **determinação da interseção dos raios** com o volume, é utilizado um algoritmo de *clipping* 3D. Para permitir uma manipulação flexível pelo usuário, o volume de raios deve ser girado, expandido e/ou contraído. Neste ambiente, torna-se fundamental descartar, eficientemente, as partes dos raios que não atingem o volume.

Como a maioria dos raios intercepta o volume, opta-se pela utilização do algoritmo de Cyrus-Beck, por ser mais eficiente na determinação dos pontos de interseção. Este algoritmo se baseia na determinação do parâmetro t , da interseção do raio com a fronteira do volume, e na classificação deste ponto como potencialmente entrando ou potencialmente saindo, de acordo com a superfície de interseção.

O valor de t é dado por:

$$t = \frac{\text{num}}{\text{den}} = \frac{-N_i * (P_0 - P_i)}{N_i * (P_i - P_0)}$$

Onde N_i é a normal da face i , P_0 é o ponto inicial do raio, P_i o ponto final do raio e, P_i é um ponto arbitrário a ser classificado como “entrada potencial” ou “saída potencial” (*figura 4.6*).

Desta forma, se $den > 0$, então, tem-se o caso “saída potencial”; se $den < 0$, tem-se o caso “entrada potencial”.

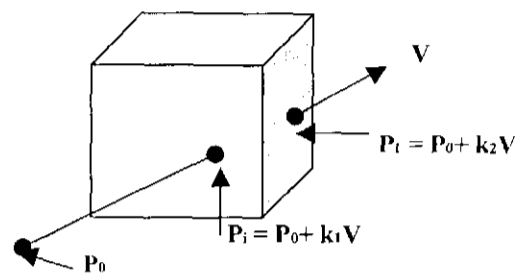


Figura 4.6. Redução do número de pontos de amostra necessários, ao longo do raio.

Também foram feitas otimizações considerando o caminhamento pelo raio, como por exemplo, quando o caminhamento é interrompido ao atingir uma superfície opaca, ou quando as contribuições deixam de ser significativas em relação aos valores já calculados, como no caso do término adaptativo dos raios.

O **término adaptativo dos raios** é uma técnica de otimização bastante simples e pode ser utilizada quando os raios são traçados desde o plano frontal para o posterior (*front to back*) [Pa97, GC91]. Isto simplesmente finaliza o desenho do raio depois que a cor acumulada para aquele raio está sobre um certo limiar.

Conforme visto anteriormente, a opacidade O_{out} de um raio, após atravessar uma amostra do volume semitransparente, é dada em função da opacidade O_{in} antes de atravessar a amostra, e da opacidade da amostra em questão:

$$O_{out} = O_{in} + O*(1-O_{in})$$

Com isto, nota-se que a opacidade é acumulada enquanto o raio atravessa o volume. Nessa situação, quando a opacidade aumenta, a cor de cada amostra tem cada vez menos influência na cor final do pixel, ou seja, $C_{out}-C_{in}$ torna-se cada vez menor. No caso limite, quando a opacidade acumulada atinge 1, a cor do pixel não sofre mais alteração, ou seja, $C_{out} = C_{in}$. Nesse caso, o traçado do raio pode ser interrompido.

Tipicamente, utiliza-se um valor limite máximo, especificado pelo usuário, para a opacidade acumulada. Quanto menor esse valor, mais rápido é o traçado de cada raio. Entretanto, quanto maior, melhor é a qualidade final da imagem.

4.5.3. Exploração da Coerência Espacial

Entre as técnicas de aceleração do *rendering* volumétrico, aquelas que atravessam de forma eficiente o espaço vazio aparecem como as mais eficientes. Estas técnicas aproveitam a idéia da uniformidade de determinadas regiões do espaço.

Dentre estes métodos, encontramos a **representação hierárquica do espaço**, que decompõe o volume em regiões uniformes, que podem ser representadas por nós em uma estrutura de dados do tipo “pirâmides”, tal como as árvores *Octree* e *BSP*, que permitem uma visualização mais rápida [Ka97].

Ao utilizar este tipo árvores, um volume tridimensional pode ser modelado dividindo a cena em cubos pequenos (células). A posição de cada cubo pode ser representada através de 3 coordenadas cartesianas (x, y, z) e, se cada cubo fosse suficientemente pequeno, poder-se-ia considerar que contém só um tipo de matéria codificado de algum modo. Porém, para grandes quantidade de dados, este poderia ser um modo ineficiente para modelar o mundo, porque uma massa grande de matéria homogênea requereria tantos dados como as partes mais complexas da estrutura exterior. Uma forma mais eficiente de manter esta informação poderia ser em uma árvore *BSP* ou em uma *Octree*.

Neste tipo de estrutura hierárquica, decompõe-se o volume de N^3 células em $\log N$ volumes, onde o primeiro é o volume original, o segundo é uma média de $2 \times 2 \times 2$ células do original, resultando em um volume com $1/8$ da resolução. Este processo se repete até serem definidos $\log N$ volumes, como no caso do algoritmo *Hierarchical Splatting* [LH91]. Uma vez construída a pirâmide, a visualização é feita percorrendo a pirâmide no nível correspondente à resolução desejada.

Uma **árvore binária de divisão do espaço** (*Binary Space Plane - BSP*) é uma estrutura de dados que representa uma subdivisão hierárquica recursiva do espaço n -dimensional, em subespaços convexos [GC91, Ba00]. A árvore *BSP* divide o espaço utilizando hiperplanos. O resultado de cada divisão são dois subespaços novos que podem ser divididos recursivamente. O

exemplo da *figura 4.7* mostra como um espaço 3D (A) pode ser dividido em regiões (B e C), e cada região (B ou C) pode ser subdividida em regiões menores (B então é dividida em D e E).

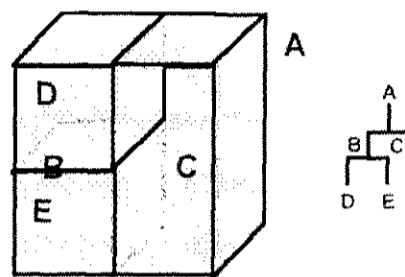


Figura 4.7. Representação da árvore BSP [Ba00].

Árvores *BSP* normalmente não são mecanismos que contêm a informação da forma do objeto. Outros métodos se encarregam de modelar a forma, como aqueles que utilizam triângulos ou polígonos. As árvores *BSP* podem ser usadas por estruturas de ordenação e classificação. São aplicadas para fazer remoção de superfície escondida ou localização de raios, hierarquias de *Ray Tracing*, modelagem de sólidos, planejamento de movimento de robôs, etc.

A **técnica de decomposição hierárquica do espaço** é uma técnica de otimização que utiliza *Octrees* ou árvores *BSP* para decompor o volume em regiões transparentes e não transparentes. Esta técnica constrói um modelo hierárquico do volume [E192, Ka97, Pa97]. Considerando-se o caso das *Octrees*, cada octante pode ser totalmente transparente (contendo somente células transparentes) ou não transparente (contendo pelo menos uma célula não transparente). A idéia principal da técnica é procurar tirar o máximo proveito da coerência espacial do volume. Com isso, durante a integração das células ao longo dos raios, pode-se descartar rapidamente regiões com células totalmente transparentes.

Inicialmente, o volume de dados constitui o nível mais alto da hierarquia. Caso exista alguma célula não transparente, o volume é subdividido em oito subvolumes (octantes ou 20 células), constituindo um nível imediatamente inferior. Se todas as células são transparentes, não é necessário subdividir. Para cada um dos oito subvolumes, o mesmo processo é aplicado até que se atinja o nível 0 da hierarquia, ou seja, não é mais possível subdividir a célula, pois ela contém exatamente oito células (nesse caso, uma célula é uma amostra do dado volumétrico).

De uma outra forma, uma célula de nível 0 é transparente (possui o valor 0) caso suas oito células dos vértices tenham opacidade igual a zero. Caso contrário, tem valor 1. Uma célula de qualquer outro nível tem valor 0 caso as oito células do nível exatamente inferior tenham valor 0. Caso contrário, esta célula tem valor 1. A *figura 4.8* ilustra a construção da *Octree* de um volume.

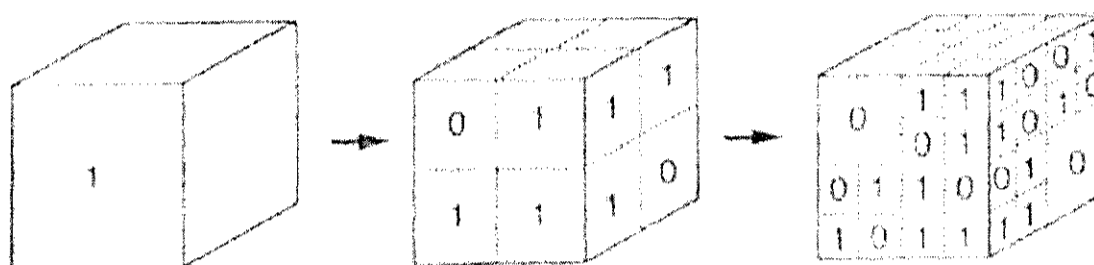


Figura 4.8. Construção de uma Octree [Le90b].

Uma vez construída a *Octree*, deve-se modificar o processo de lançamento de raios do algoritmo *Ray Casting*. Para cada raio lançado, primeiro é calculada a interseção do raio com a célula de nível básico (que contém todo o volume). Se ela contém 0, pode-se avançar para a próxima célula do mesmo nível. Se o pai desta nova célula não for o mesmo da célula anterior, deve-se subir um nível na hierarquia. Isso é muito importante caso essa nova célula contenha 0. Nessa situação, o algoritmo economiza tempo, pois avança rapidamente pelo raio, ignorando todas as células totalmente transparentes.

Essencialmente, a idéia é avançar até atingir uma célula contendo 1. Nesse caso, deve-se descer um nível na hierarquia. Se ao descer se atinge o nível 0, sabe-se que pelo menos uma das oito células dessa célula possui opacidade maior que 0. Nessa situação, deve-se amostrar o raio ao longo da célula e calcular a contribuição das amostras para a cor final do pixel, como acontece no algoritmo padrão.

4.5.4. Acumulação das Contribuições de Opacidade e Cor ao Longo do Raio

A acumulação das contribuições de cor e opacidade para cada célula é feito durante o traçado do raio por pontos de amostragem equidistantes. No entanto, este procedimento possui como inconveniente a determinação da célula correspondente a cada amostragem.

Para contornar tal inconveniente, propõe-se geralmente a utilização do algoritmo de Bresenham, originalmente desenvolvido para desenhar linhas retas em um dispositivo matricial, permitindo uma forma incremental de caminhar no raio, além de usar somente aritmética inteira. Desta forma, evita-se o problema de localização e garante-se que, para um mesmo raio, não existam duas amostragens para uma mesma célula.

Além disso, deve-se considerar o modelo de iluminação utilizado pelos diferentes algoritmos. Tipicamente, é utilizado um modelo de iluminação local, mas pode-se utilizar modelos de iluminação global como o proposto em [YCK92].

4.5.5. Exibição do Plano de Visualização Durante a “Construção” da Imagem

O **refinamento progressivo** é utilizado para evitar o desperdício de memória e obter uma maior interatividade com o usuário. Apresentar a imagem em refinamentos sucessivos é uma alternativa prática de solução para o problema do tempo de *rendering* através da geração de imagens parciais que são refinadas progressivamente, assim que o usuário interage com a imagem inicial.

A idéia é percorrer uma grade regular sobre o plano de visualização, diminuindo, a cada etapa, o passo na grade [Le90a, Le90b, Le90c, SGDM95]. Assim, a imagem é mostrada rapidamente em baixa resolução, sendo refinada até a resolução final. Como ilustra a *figura 4.9*, cada pixel da imagem é calculado uma única vez. Portanto, o tempo de cálculo da cor dos pixels permanece inalterado. Assim, obtemos o mesmo efeito pretendido quando na utilização de “pirâmides”, sem nenhuma memória adicional. Embora não seja estritamente uma técnica de otimização, esta retroalimentação (*feed-back*) imediata permite que o usuário interrompa o processo de visualização para, por exemplo, escolher outro ponto de visão.

Quando o processo de exibir um pixel é suficientemente rápido, a ordem dos pixels não é importante. Entretanto, se o processo de exibição for lento ou se a resolução da imagem for alta, o tempo de exibição pode ser muito alto, o que é tedioso para o usuário.

A *figura 4.9a* ilustra a idéia de percorrer uma malha 2D regular sobre a área da imagem, diminuindo a cada instante, o passo na malha. A *figura 4.9b* ilustra uma técnica simples de refinamento progressivo. A imagem é mostrada como se tivesse baixa resolução, sendo refinada até conseguir a imagem final. Cada pixel é examinado uma única vez.

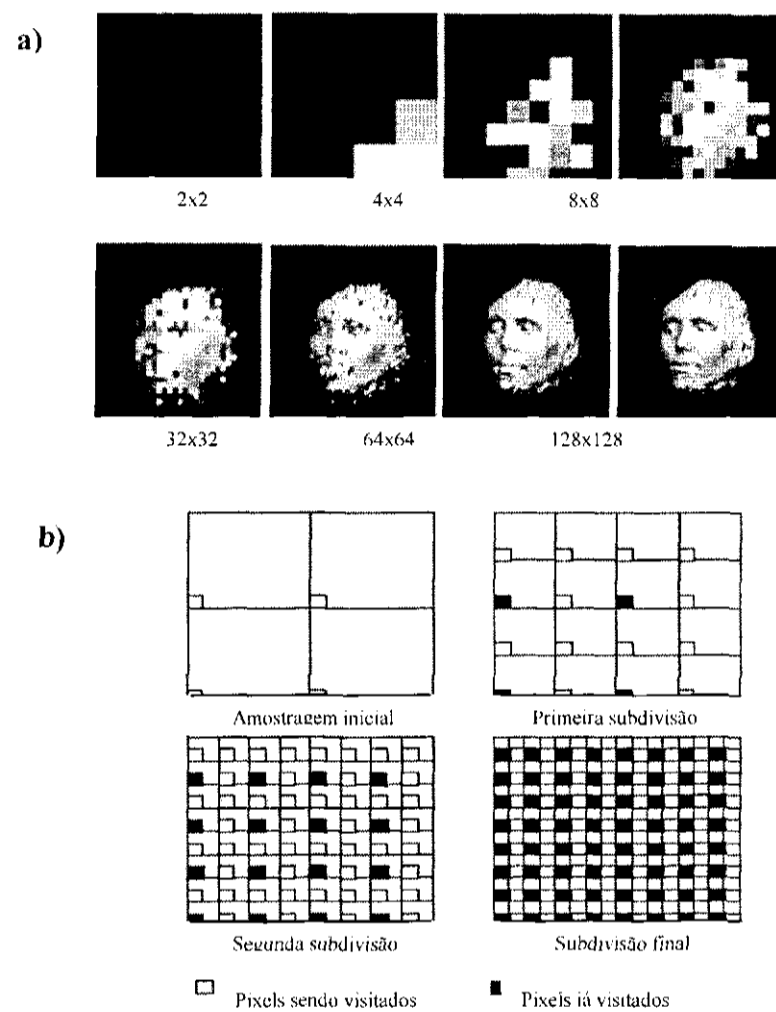


Figura 4.9. Construção da imagem: a) Vistas prévias do refinamento b) Técnica de progressão da imagem [SGDM95].

4.5.6 Aceleração Utilizando Algoritmos Paralelos

A quantidade de dados volumétricos a ser representado, mesmo em baixas resoluções, é enorme. Uma das soluções mais óbvias para reduzir o tempo total de *rendering* é utilizar algoritmos e arquiteturas paralelas.

Esta abordagem é particularmente importante se considerarmos o fato de que muitas simulações numéricas que produzem dados volumétricos interessantes são feitas em arquiteturas paralelas, podendo produzir quantidades muito grandes de dados, sendo inconveniente juntar os dados a serem visualizados em *workstations*, após serem processados [Ch93].

As arquiteturas paralelas podem ser classificadas seguindo diferentes parâmetros. Um desses parâmetros é a autonomia dos processadores [Ch92]. Segundo este parâmetro, as arquiteturas, tipicamente, se dividem em dois grupos: sistemas onde os processadores executam uma mesma instrução, em um determinado momento sobre um grande conjunto de dados (denominadas arquiteturas SIMD – “*Single Instruction Multiple Data*”); e sistemas onde cada processador execute um conjunto de instruções diferentes sobre múltiplos dados (denominadas por isso arquiteturas MIMD – “*Multiple Instruction Multiple Data*”).

Uma outra classificação dos sistemas é pela arquitetura da memória. A memória pode ser totalmente compartilhada, mas limitada pela amplitude da faixa de dados; totalmente distribuída e privada, tendo-se que comunicar mediante mensagens em uma rede interconectada; ou uma combinação destas duas.

Uma terceira classificação depende do tipo de interconexão da rede utilizada, podendo estar associados os processadores e memória sob um sistema conectado mediante uma arquitetura de bus, anel (*ring*), malha, hipercubo, ou rede multiestado de *switching*. Referências a estas arquiteturas podem ser encontradas em [Ch93, Ka97, SK95].

Algumas das implementações mais recentes dentro do *rendering* volumétrico direto foram feitas em uma variedade de arquiteturas paralelas, algumas delas sobre processadores paralelos experimentais, tais como DASH [NL92], the Princeton Engine [SS92], PARCUM II [Ja88], entre outras. Algumas outras implementações foram realizadas em máquinas paralelas comerciais como a Connection Machine CM-2 [SS91], nCUBE [KB88], Network of Suns [We90], só para mencionar algumas.

Originalmente suportavam só implementações para malhas regulares baseadas em *voxels* [Si95, Le90c, SK95], mas depois foram desenvolvidos algoritmos paralelos de *Ray Casting* para malhas irregulares, como por exemplo, para malhas curvilíneas encontramos o algoritmo descrito em [Ga92], para malhas não estruturadas em [Ga93, Ma95, Pr96], e inclusive para formatos de dados irregulares comprimidos em [CTT00]

4.6. Considerações Finais

Neste capítulo apresentaram-se as bases do algoritmo de *Ray Casting* para a visualização de volumes, uma visão geral das técnicas desenvolvidas utilizando este algoritmo, e algumas das principais otimizações feitas nele.

Após uma breve introdução, foi apresentada uma visão global de alguns dos principais trabalhos correlatos baseados no *Ray Casting*. Foi, então, definido o primeiro método que utilizou o conceito de *Ray Casting* conhecido como *Ray Casting* Binário. Em seguida, apresentou-se uma descrição geral e intuitiva do método de *Ray Casting* desenvolvido por Levoy, que é a base dos seguintes trabalhos e otimizações feitos utilizando esta técnica, e também, foram descritas as etapas de classificação e iluminação deste modelo.

Uma seção foi dedicada à apresentação das diferentes abordagens utilizadas para otimizar o *Ray Casting* em suas diferentes etapas: lançamento dos raios, determinação dos segmentos dos raios que interceptam o volume, exploração da coerência espacial do volume, acumulação das contribuições de opacidade e cor ao longo do raio, exibição do plano de visualização durante sua construção, e, considerando-se também como uma otimização, a aceleração utilizando algoritmos paralelos.

Este capítulo foi dedicado exclusivamente ao algoritmo de *Ray Casting*, por este constituir a base da proposta apresentada neste trabalho, para o *Rendering* Volumétrico de Malhas Não Estruturadas. Particularmente, o *Ray Casting* foi escolhido por permitir uma visualização de forma natural, obter imagens de alta qualidade, e permitir a visualização de estruturas internas através do uso de transparência nas estruturas mais externas. A variedade de estudos anteriores feitos sobre este algoritmo permite, tanto um melhor entendimento dos fundamentos nos quais se baseia, quanto aproveitar as idéias de otimização que podem ser adotadas para permitir uma melhor e mais rápida visualização.

CAPÍTULO 5

Uma Técnica Híbrida de Visualização para a Exploração de Dados Volumétricos Não Estruturados

5.1. Considerações Iniciais

O alvo deste trabalho é o desenvolvimento de uma técnica de visualização volumétrica para uso em ambientes interativos. A nova técnica proposta aqui, além de permitir interatividade, pode ser aplicada a malhas não estruturadas, como será mostrado nos exemplos. Esta técnica está baseada em uma estratégia híbrida de *rendering* superficial e volumétrico, e pode ser facilmente paralelizável quando empregada em uma arquitetura de memória compartilhada.

A integração do RS com RVD não é uma tarefa fácil, mas é uma característica desejável para muitas aplicações. Assim, a técnica de *rendering* proposta proporciona uma pré-visualização direta na superfície do volume o que permite acelerar as transformações de projeção, viabilizando uma maior interação com o usuário.

Além disto, esta técnica de *rendering* híbrido utiliza uma forma geral de representação mediante malhas não estruturadas. Ao utilizar este tipo de malhas, devemos considerar que, tanto a configuração geométrica destas malhas, como os cálculos matemáticos dos modelos de iluminação do *rendering*, são mais complexos que em decomposições regulares [TL88]. Devido às vantagens já apresentadas anteriormente, o tetraedro foi escolhido como a unidade de representação nas implementações realizadas nesse trabalho. Como em outros sistemas de *rendering* volumétrico para malhas não estruturadas [NSG90, YCK92, JC94, We94, Max95,

KSS00, NMS00], a técnica apresentada aqui utiliza várias aproximações com o propósito de simplificar o processo de visualização. A estratégia de visualização proposta é dividida basicamente em três etapas, que são sumarizadas a seguir.

Primeiramente, os dados volumétricos são armazenados utilizando uma estrutura de dados topológica que representa as informações de incidência e adjacência.

A segunda etapa do processo consiste em aplicar uma técnica de RVD. O RVD utilizado é uma versão modificada do algoritmo clássico de *Ray Casting* apresentado em [Le88, Le90a]. A idéia geral deste novo *Ray Casting* é tomar como plano de pré-projeção cada uma das faces do bordo do volume armazenado. A partir de cada face é lançado um conjunto de raios – cada raio com uma direção diferente – através do volume. Para cada raio são calculadas as interseções com as células tetraedrais e tomadas amostras de cor e opacidades para compor um escalar que será associado com o raio. O raio atravessa a malha tetraedral ate atingir uma outra face de bordo ou a opacidade alcança um limiar determinado pelo usuário.

Concluída a etapa do *Ray Casting*, realiza-se a última etapa do *rendering* onde é aplicado um método de RS. Nessa ultima etapa somente os elementos de bordo do volume onde a pré-projeção foi realizada são considerados. A partir da posição do observador, utiliza-se um dos escalares associados com os raios lançados das faces de bordo para definir a cor dessa face. Dessa forma, para cada face adota-se a cor associada a um dos raios de luz. Será eleito o raio com direção mais próxima à direção do observador. Pode-se suavizar a imagem utilizando um modelo de iluminação local para a superfície de bordo. A utilização do RS para a projeção final permite ao usuário navegar pela cena volumétrica em tempo real e principalmente fazer uso de *hardware* e *software* comuns, como as placas gráficas e *OpenGL*.

5.2. Pipeline de Visualização Híbrido

Nesta seção descreveremos detalhadamente cada uma das três etapas do *rendering* híbrido descrito resumidamente na seção anterior. Como visto acima, a estratégia de *rendering* é dividida em:

- Primeira etapa: Armazenamento dos dados brutos e da malha tetraedral;

- Segunda etapa: Aplicação de um modelo de *rendering* volumétrico direto baseado no *Ray Casting*.

- Terceira etapa: Aplicação de uma técnica de *rendering* superficial, para as faces de bordo da malha tetraedral.

O diagrama da *figura 5.1* resume o *pipeline* de visualização desta técnica híbrida.

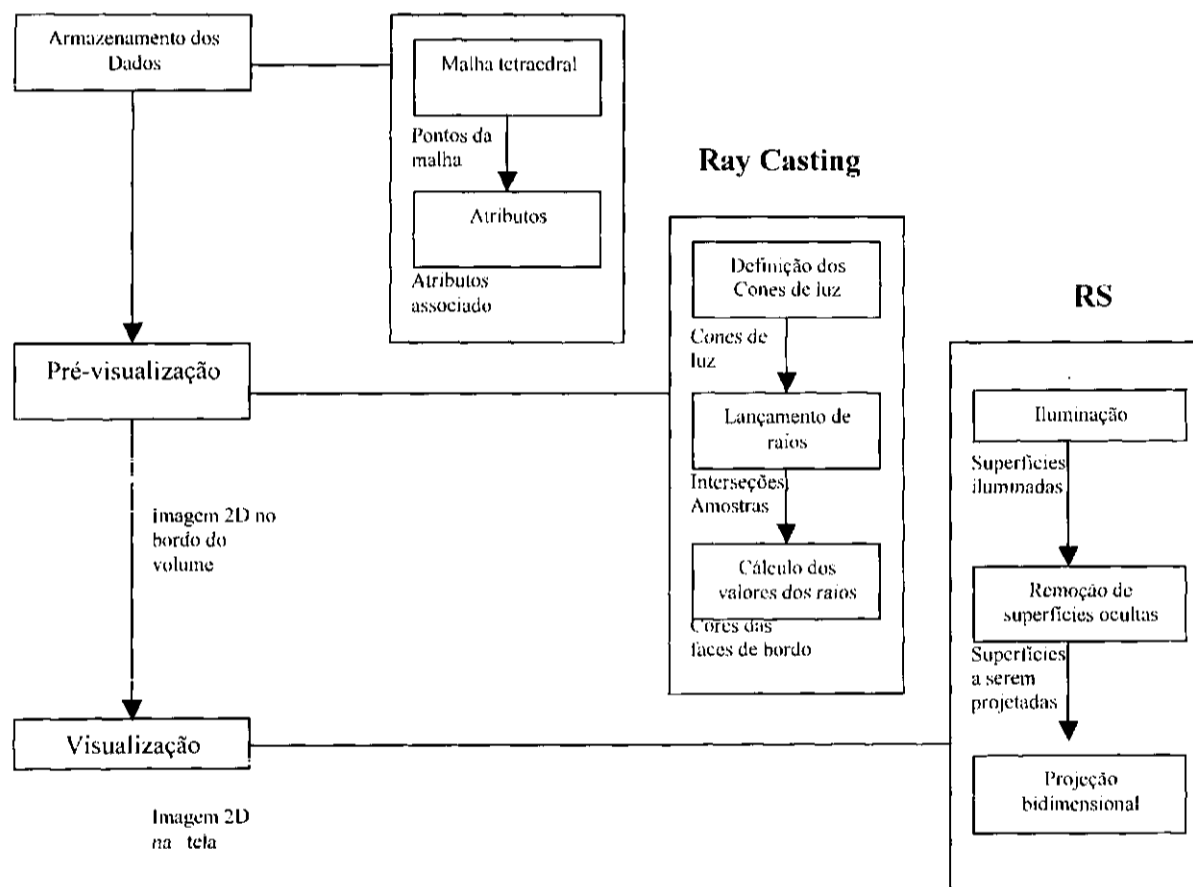


Figura 5.1. Pipeline de Visualização Híbrida.

Como em outras técnicas baseadas em lançamento de raios, a relação topológica entre as células é fundamental para um cálculo eficiente de cor e opacidade ao longo dos raios. No caso

das malhas não estruturadas, um aspecto importante é solucionar o problema da complexidade das interseções dos raios com as células [BCL99], nos casos especiais, onde os raios são paralelos a uma face, ou passam por uma aresta ou vértice. Vale reforçar que a restrição às células tetraedrais simplifica os cálculos nesses casos.

As particularidades de cada uma das etapas do processo de *rendering* híbrido são detalhadas a seguir.

5.2.1. Armazenamento dos Dados Volumétricos

O *rendering* de malhas irregulares tem sido identificado como uma área de pesquisa especialmente importante na visualização [Ka94]. Em geral as técnicas de *rendering* volumétrico para malhas não estruturadas utilizam uma ordenação das células para calcular a ordem de projeção dos elementos no plano de visualização [MHC90, SM97].

Neste trabalho será considerado que a malha de entrada é armazenada em uma estrutura de dados topológica contendo as informações de incidência e adjacência para cada elemento da malha. Desta forma é possível percorrer as faces e elementos vizinhos de cada célula de forma imediata. Utilizamos como entrada arquivos do tipo *unstructre grid* do VTK, que já contém informação de incidência entre elementos. No caso onde essa relação não é dada, é necessário realizar um pré-processamento para organizar os dados.

A estrutura utilizada na implementação do RV híbrido é bastante simples e representa basicamente as relações de vizinhança necessárias para uma implementação eficiente do *Ray Casting*. Particularmente, armazena-se com cada vértice v , informação de sua estrela, que é uma lista das células da malha que contém v (v é um vértice da célula). Armazena-se também uma lista de ponteiros para faces de bordo do volume (importante para as etapas posteriores do algoritmo) e as relações de incidência entre as faces de tetraedros vizinhos.

O espaço da imagem, que normalmente consiste de $N \times N$ pixels, para o caso do processo de *rendering* direto será substituído pelas M faces de bordo dos objetos da cena.

Além de serem considerados os dados da malha a estrutura também armazena os valores dos seus atributos. Os atributos característicos são cor e opacidade. O usuário pode relacionar diretamente estes atributos com cada dado do volume, ou pode agrupa-los sob uma mesma

denominação como, por exemplo, materiais, o que pode caracterizar determinadas regiões. Nesta etapa podem ser utilizadas diferentes funções de transferência, de forma similar a classificação mencionada na *seção 4.4.1*.

5.2.2. Ray Casting

Como mencionado, a base do *rendering* híbrido desenvolvido neste trabalho é o *Ray Casting*, não só por ser uma das técnicas de visualização volumétrica direta mais difundidas e conhecidas, o que facilita seu entendimento, mas também pelas características e vantagens que apresenta sobre outros algoritmos de visualização em malhas não estruturadas.

A variação do *Ray Casting* utilizada, baseia-se em lançamento de raios a partir de um ponto em cada face de bordo do volume, atravessando o volume. Dessa forma, cada face de bordo é considerada como plano de projeção, não levando-se em consideração os pixels do plano de visualização. Para cada raio lançado, uma amostragem é feita com base nos valores das células. A cor final de cada face do bordo é obtida integrando as contribuições de cor e opacidade de cada célula interceptada pelo raio.

Previamente ao lançamento dos raios, deve-se definir cones de raios – também denominados cones de luz – que organizam as direções de lançamento dos raios. A forma como os cones de luz são construídos, será explicada mais a frente.

Como foi observado no *pipeline* de visualização (sumarizado aqui), a etapa de *Ray Casting* utilizada por esta técnica híbrida pode ser dividida em três módulos:

- Definição dos cones de luz;
- Lançamento dos raios e interseções;
- Cálculo dos valores para cada raio.

Se necessário, pode-se acrescentar dois módulos prévios aos mencionados: um de classificação e outro de iluminação dos dados volumétricos. A classificação é feita da mesma forma que para qualquer outra técnica de *rendering* que utiliza *Ray Casting*, como definida na *seção 4.4.1*. Uma vez que os dados foram classificados, pode-se também aplicar um modelo de iluminação sobre eles.

O modelo de iluminação é utilizado para criar a ilusão de profundidade, realçar bordas e características do volume na visualização. Foi considerada aproximações de modelos de iluminação de baixa especularidade, pois computacionalmente são mais simples e apresentam bons resultados, especialmente para imagens obtidas mediante técnicas não invasivas, como as tomografias computadorizadas. A utilização de modelos com alta especularidade proporcionam maior realismo, porém, em casos como este, em que se utiliza transparência para a análise de estruturas internas, não traz grandes benefícios. Os modelos de baixa especularidade utilizados consideram a utilização de uma ou múltiplas fontes de luz. Tipicamente, a luz emitida pelas fontes presentes na cena não é atenuada pelo volume. Este efeito é ignorado durante o cálculo da iluminação, pois, além de simplificar bastante o algoritmo, evita-se que áreas ao redor de regiões muito densas fiquem totalmente escuras. Entretanto, o efeito da atenuação da luz ao longo do raio de visão não pode ser desprezado. Isso é levado em consideração durante o cálculo dos valores dos raios.

Definição dos cones de luz

O *Ray Casting* está baseado no lançamento de raios a partir de um ponto definido sobre cada face de bordo, neste caso, o baricentro que é calculado previamente.

Uma vez determinado o ponto de partida do raio para uma face, é necessário determinar a direção que esse raio seguirá. A direção de lançamento inicial é determinada pela direção inversa da normal da face, que em nossa representação, está orientada para fora da célula.

A cada face está associado um cone de raios que possui como eixo o vetor normal da face. A partir do cone de luz é possível definir um conjunto de direções de lançamento de raios (veja *figura 5.2*). A “abertura” do cone é o número de raios em cada cone é definido pelo usuário.

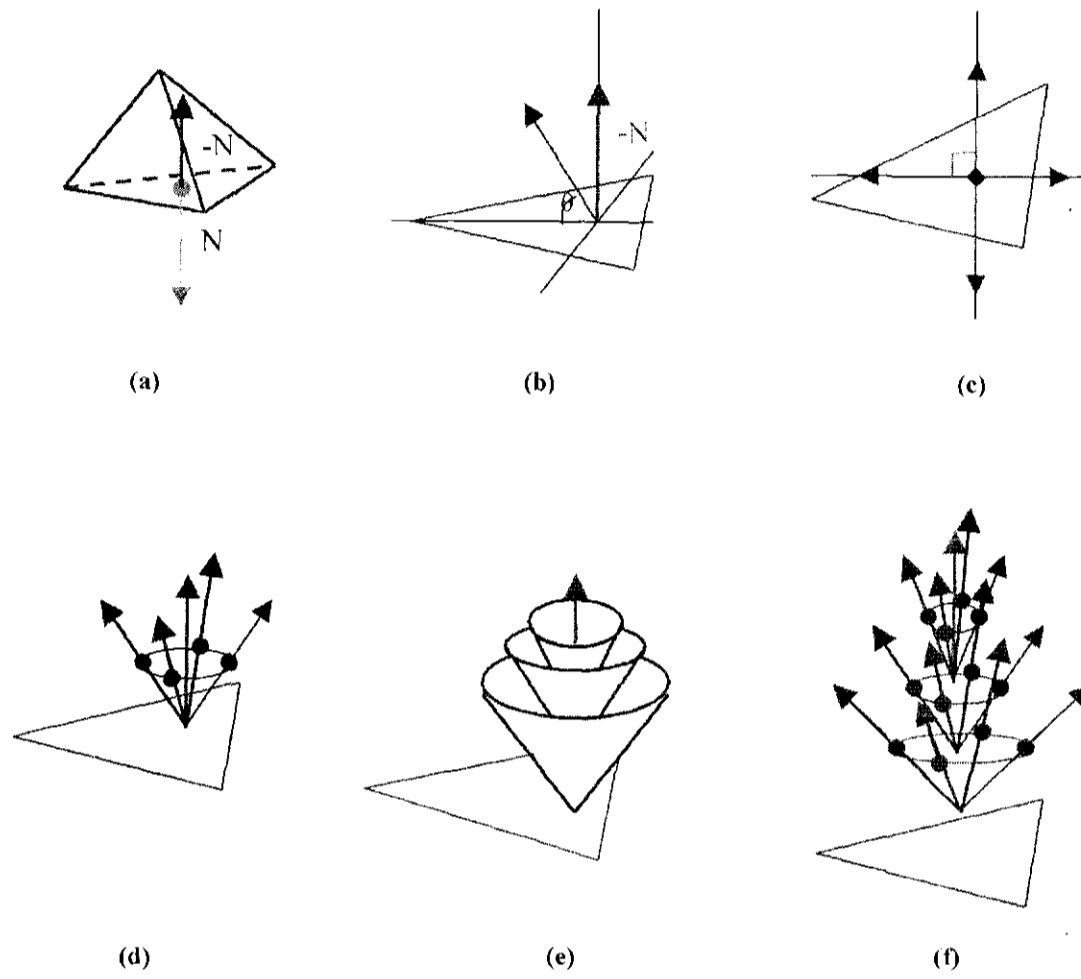


Figura 5.2. Cone de luz: a) Determinação da direção primaria, b) Determinação do primeiro raio do cone, c) Determinação dos outros raios, d) Cone final de raios, e) Três cones, f) Raios dos três cones anteriores.

Lançamento dos raios e interseções

Uma vez lançado um raio a partir de um cone de luz, são calculadas as interseções desse raio com as células da malha (figura 5.3). A primeira célula é atravessada e utiliza-se a informação de vizinhança armazenada nela para identificar qual é a célula seguinte no caminho do raio.

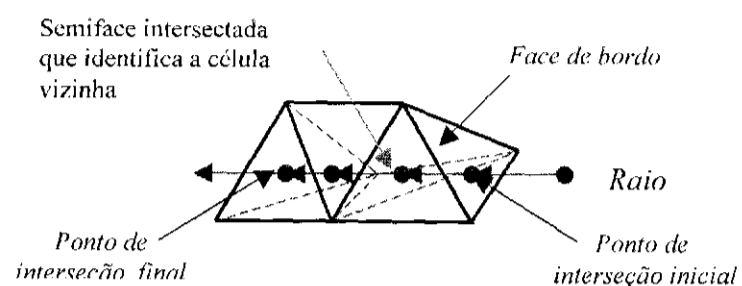


Figura 5.3. Cálculo das interseções de um raio.

Uma vez que é identificada uma interseção, pode-se enfrentar um dos três casos:

a) O ponto de interseção se encontra dentro de uma face (*figura 5.4a*). Este é o caso mais simples de todos, pois a relação de incidência entre as faces de tetraedros vizinhos permite detectar rapidamente qual é a próxima célula a ser analisada para a determinação da próxima interseção. Se uma face não possui informação de incidência sabe-se que ela é uma face de bordo e o raio pára nesse instante.

b) O ponto de interseção se encontra no meio de uma aresta (*figura 5.4b*). Este é um caso especial, pois é mais difícil determinar qual será a próxima célula a ser analisada no percurso do raio. Para determinar qual será a próxima célula atravessada pelo raio, utiliza-se as estrelas dos vértices da aresta, que possibilitam navegar pelos tetraedros incidentes. Para cada tetraedro incidente na aresta testa-se o raio contra suas faces. Se o raio não intersecta nenhuma delas, a procura continua até encontrar uma interseção ou verificar que a aresta é de bordo. Problemas podem acontecer quando a aresta se encontra na mesma linha de direção do raio, ou o raio intersecta uma aresta e continua no mesmo plano que uma das faces que contem a aresta. Para esses casos é utilizada uma perturbação numérica que, em geral, resolve o problema.

c) O ponto de interseção coincide com um dos vértices da célula que está sendo atravessada pelo raio (*figura 5.4c*). Este é um outro caso especial, pois é ainda mais difícil determinar qual será a próxima célula a ser analisada. Para determinar qual é a próxima célula que será atravessada pelo raio, utiliza-se também a estrela do vértice em questão. Uma a uma, as células da estrela são testadas para ver se são intersectadas pelo raio, isto requer um maior número de testes que os dois casos anteriores. Se nenhuma das células da estrela for atravessada, então, o algoritmo pára por ter achado um vértice que está no bordo do objeto.

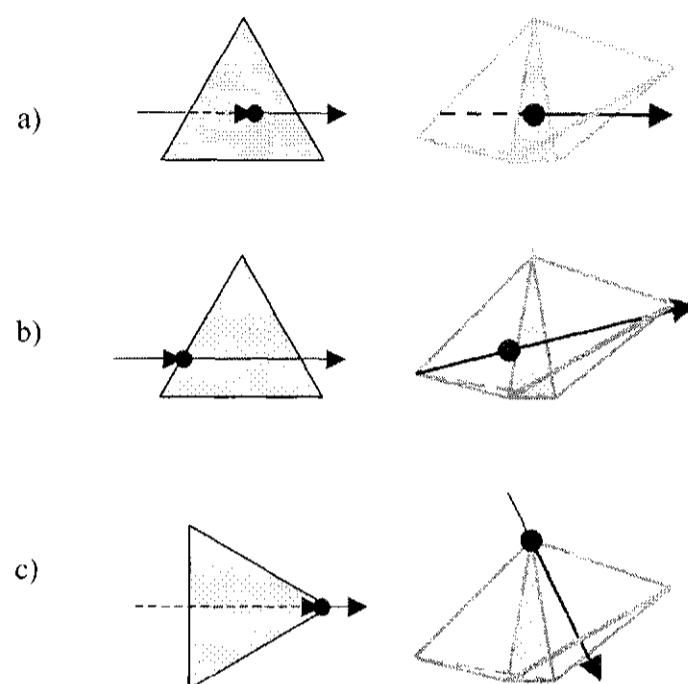


Figura 5.4. Casos de interseção do raio com uma face: a) interseção dentro da face, b) interseção com uma aresta da face, c) interseção com um vértice da face.

Cálculo dos valores para cada raio

Para cada raio, é realizada uma amostragem do volume, sendo que os intervalos entre as amostras não são constantes ao longo do raio, pois depende do ponto de interseção do raio com as faces, como mostra a *figura 5.5*.

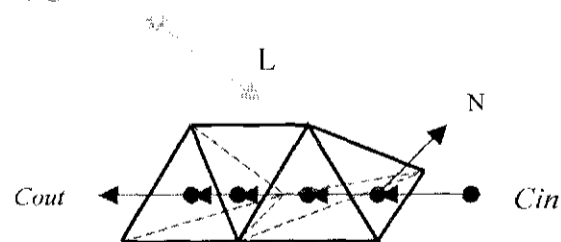


Figura 5.5. Cálculo de iluminação nas células.

A implementação considera que o volume não atenua a luz emitida pelas fontes de luz presentes na cena, pois esse fato simplifica bastante o algoritmo (não é necessário calcular uma integral de linha) e evita problemas de escurecimento de regiões onde a densidade de elementos é grande. Entretanto, o efeito da atenuação da luz ao longo dos raios lançados não pode ser desprezado, visto que é levado em consideração durante a etapa do cálculo da cor.

Por meio de interpolação dos valores armazenados nos vértices, é obtida a cor e a opacidade em cada ponto amostrado (figura 5.6). Todas as contribuições obtidas de cores e opacidades são compostas, calculando-se, assim, a cor final $C(R)$ que será associada ao raio.

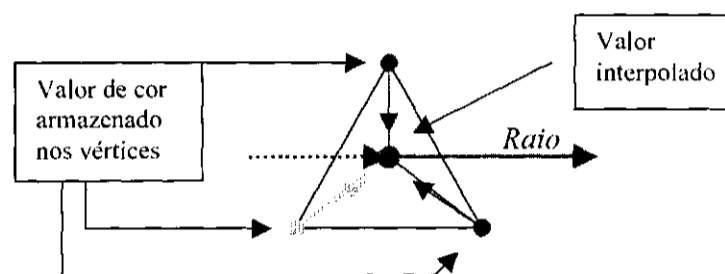


Figura 5.6. Obtenção do valor do ponto amostrado mediante interpolação.

A cor $C_{out}(R,k)$ e a opacidade $O_{out}(R,k)$ do raio, após passar pela amostra k , são calculadas em função da cor $C_{in}(R,k)$ e da opacidade $O_{in}(R,k)$ e obtidas anteriormente ao longo do raio, e também em função da $C(R,k)$ e opacidade $O(R,k)$ calculada na amostra, ou seja,

$$C'_{out}(R,k) = C'_{in}(R,k) + C'(R,k) * (1 - O_{in}(R,k))$$

$$O_{out}(R,k) = O_{in}(R,k) + O(R,k) * (1 - O_{in}(R,k))$$

Sendo que,

$$C'_{in}(R,k) = C_{in}(R,k) * O_{in}(R,k)$$

$$C'_{out}(R,k) = C_{out}(R,k) * O_{out}(R,k)$$

Como se observa na *figura 5.7*, após o cálculo de todas as contribuições, a cor final $C(R)$ associada ao raio é dada por:

$$C(R) = C'_{out}(R,N) / O_{out}(R,N)$$

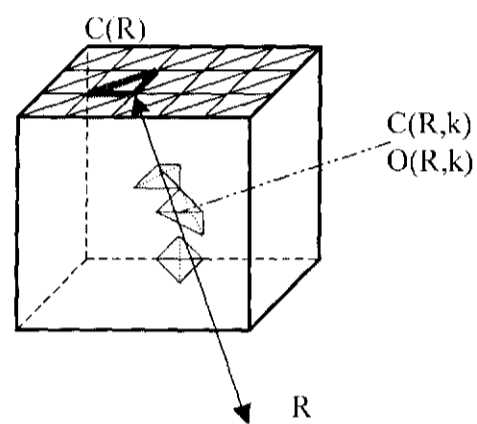


Figura 5.7. Composição de cor e opacidade.

Essencialmente, esse cálculo pode ser expresso como:

$$C(R) = \sum_{k=1}^N \left[C(R,k) * \prod_{i=1}^{k-1} (1 - O(R,i)) \right]$$

É fácil perceber que o custo do algoritmo varia dependendo da quantidade de raios a serem lançados pelo volume, e que as relações de incidência armazenadas melhoram o desempenho do algoritmo.

O processo de composição do raio também é acelerado realizando-se a composição durante a trajetória do raio, pois se pode interromper o processo quando a opacidade acumulada atinge um certo valor limite.

Como os raios lançados são totalmente independentes entre si, pode-se paralelizar facilmente esta etapa, para conseguir diminuir ainda mais o tempo de pré-processamento.

Ao pré-processar todas as faces de bordo e armazenar as informações de cor dos raios na estrutura de dados do volume, consegue-se, em uma etapa posterior de visualização, aproveitar as vantagens da visualização por superfícies.

5.2.3. *Rendering* de Superfícies

Nesta etapa final são feitos os cálculos do *rendering* de superfícies para a cena volumétrica, que permite uma visualização bastante rápida. Para acelerar ainda mais este processo, o *rendering* das primitivas geométricas (faces) pode ser feito diretamente pelo *hardware* disponível em computadores com o uso de placas gráficas comuns.

A cor de cada face é determinada da seguinte forma: encontra-se o vetor do cone de luz da face, que forma o menor ângulo com a posição do observador, e toma-se a cor associada a esse vetor como a cor da face (*figura 5.8*).

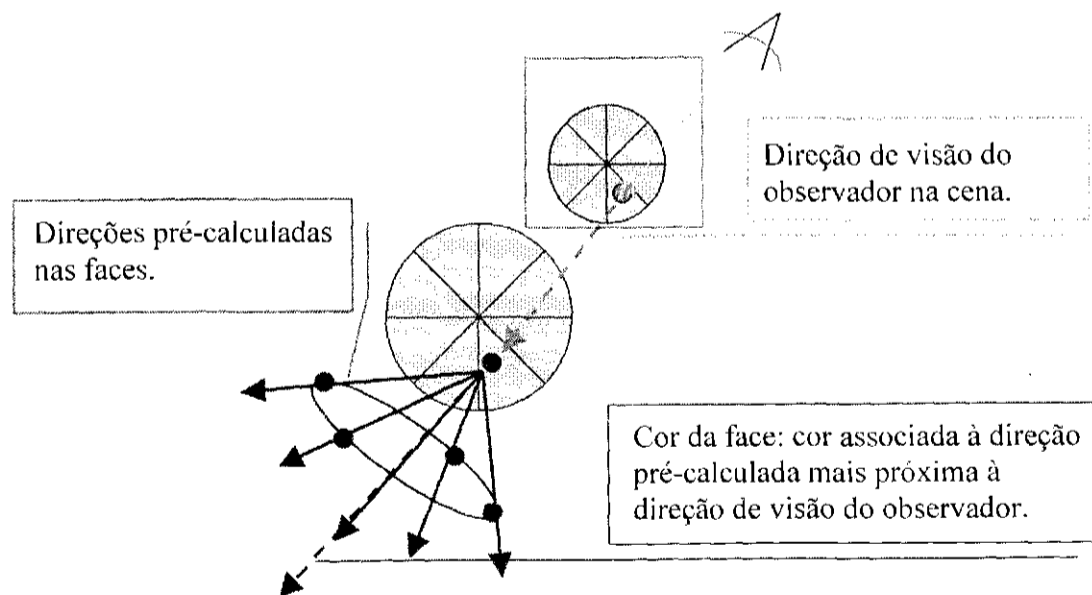


Figura 5.8. Determinação da cor da face.

Uma vez que foram definidas as cores de cada face, a cena está pronta para ser projetada. Para obter imagens mais suaves pode-se aplicar um modelo de iluminação sobre as superfícies dos objetos da cena, tipicamente utiliza-se o modelo de iluminação de Phong [PT75].

5.3. Implementação

Na implementação da técnica descrita neste trabalho, foi utilizada a linguagem C++ e a biblioteca gráfica *OpenGL*.

Como mencionado, para o armazenamento dos dados, utiliza-se uma estrutura de dados topológica que também foi implementada em C++. Esta estrutura de dados está representada na *figura 5.9*.

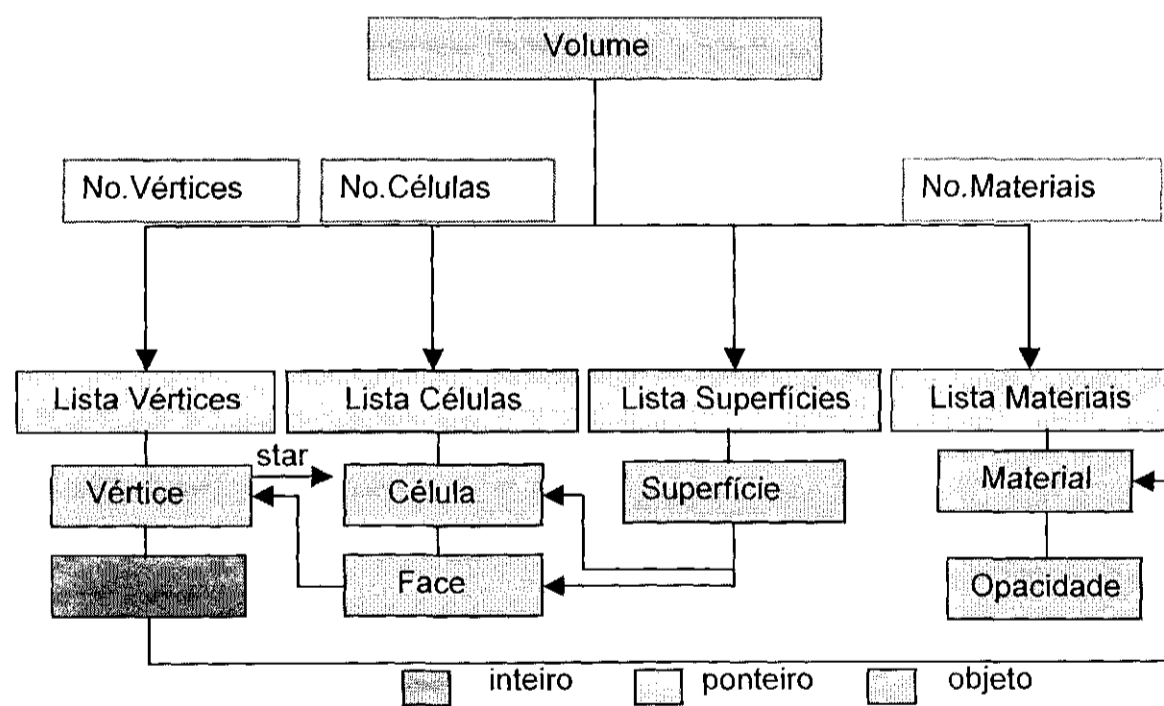


Figura 5.9. Esquema da Estrutura de Dados.

Uma das características desta estrutura de dados é a existência de uma estrela do vértice, para cada vértice v , que é uma lista das células que compartilham v . Além, da estrutura vértice considera-se a estrutura face f , cuja característica principal é permitir avançar para outra célula vizinha armazenando qual outra célula compartilha esta mesma face mediante um ponteiro.

O volume armazenado tem uma lista de ponteiros a suas superfícies (célula e sua face de bordo). Isto é útil na hora do cálculo da pré-imagem, no qual serão determinados os valores de cor e opacidade da pré-imagem, para cada face.

A segunda técnica, utilizada para o cálculo do *Ray Casting*, é a amostragem direta. Isso significa que, na hora de calcular o ponto de intersecção entre as faces das células e o raio é feita a amostragem de cor e opacidade para o cálculo discreto da cor final do raio.

A terceira técnica utiliza um limiar de opacidade para conseguir uma terminação adaptativa do seguimento do raio. Uma finalização rápida de um raio reduz o número de células que devem ser amostradas, e o número de comparações pra achar as intersecções do raio com as células.

A quarta técnica consiste no lançamento de raios a partir das faces de bordo dos objetos, e não de cada pixel, sendo utilizado um número pré-determinado de raios por cada face, para diferentes direções de visão.

A escolha dos ângulos, que determinarão as direções dos raios assim como o número de raios lançados desde cada face, é um fator importante para a obtenção de boas imagens, pois se deve procurar manter um bom equilíbrio entre a qualidade da imagem que se quer obter e o custo de memória que requerem, tanto o cálculo de um determinado número de raios por face, quanto do seu armazenamento para posteriores transformações.

Uma vez que foram lançados todos os raios para todas as faces de bordo de todos os objetos, a cena está pronta para passar à etapa de *rendering* de superfícies.

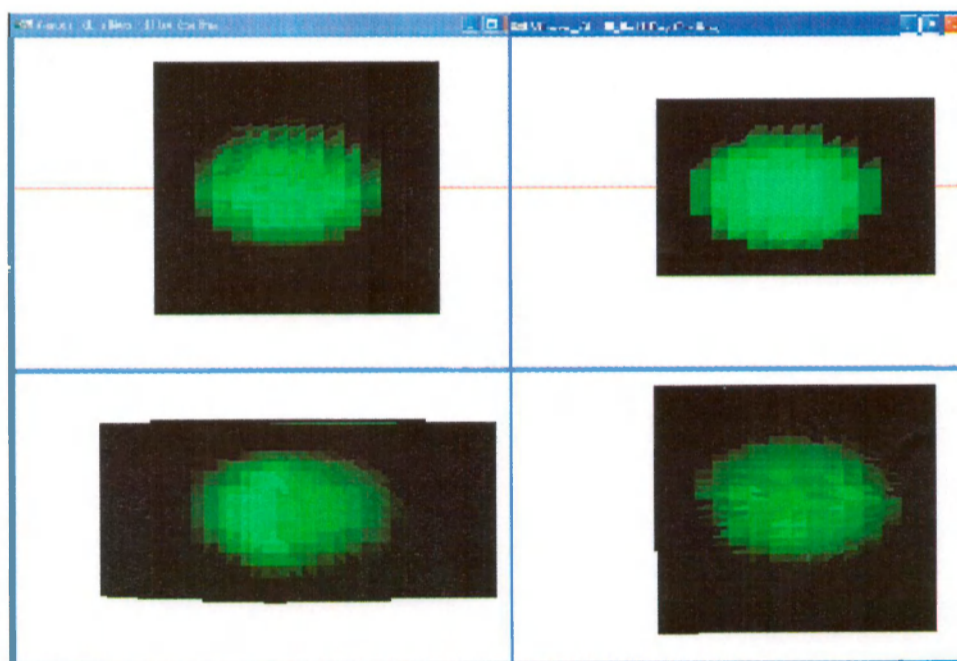
Durante a etapa de *rendering* de superfícies, dado o fato de que as faces das superfícies a ser visualizadas já são conhecidas, pois foram identificadas em etapas anteriores, só é necessário manter armazenada a informação destas faces.

Cada vez que o observador muda seu ponto de visão, transformações são aplicadas nos dados. Estas transformações são feitas, tanto sobre os pontos ou vértices, como sobre as direções que serão comparadas para achar aquela mais próxima ao novo ponto e/ou direção de visão do observador.

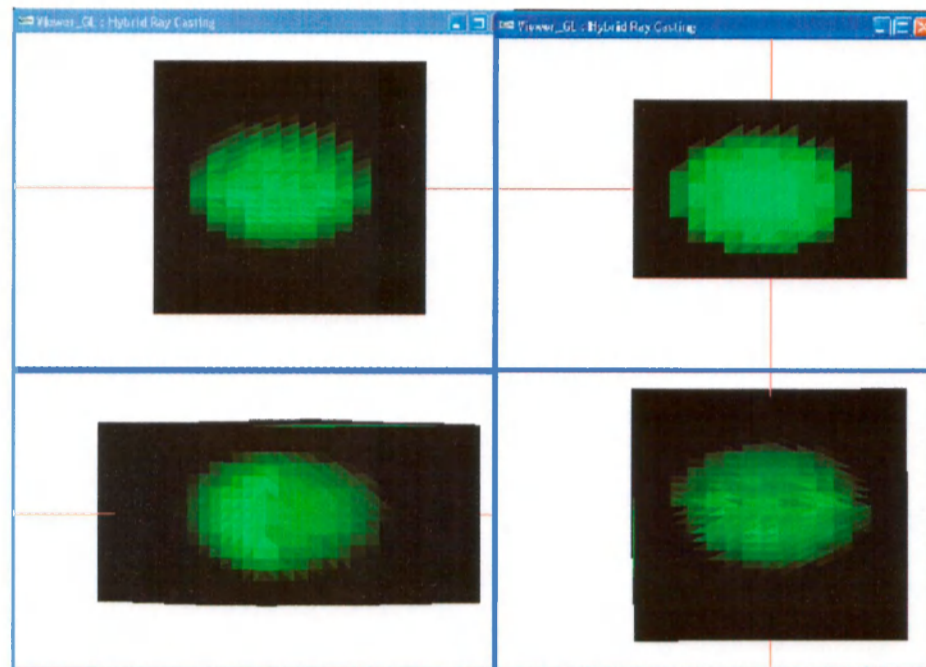
Este ambiente de visualização volumétrica híbrida facilita as transformações sobre as superfícies dos objetos, e aproveitar melhor os diferentes recursos gráficos do computador, o que nos leva a conseguir uma visualização em tempo real após o pré-processamento feito pelo *Ray Casting*. Isto permite ao observador navegar pela cena e, interagir com o modelo.

A seguir são apresentadas imagens geradas com a técnica híbrida descrita:

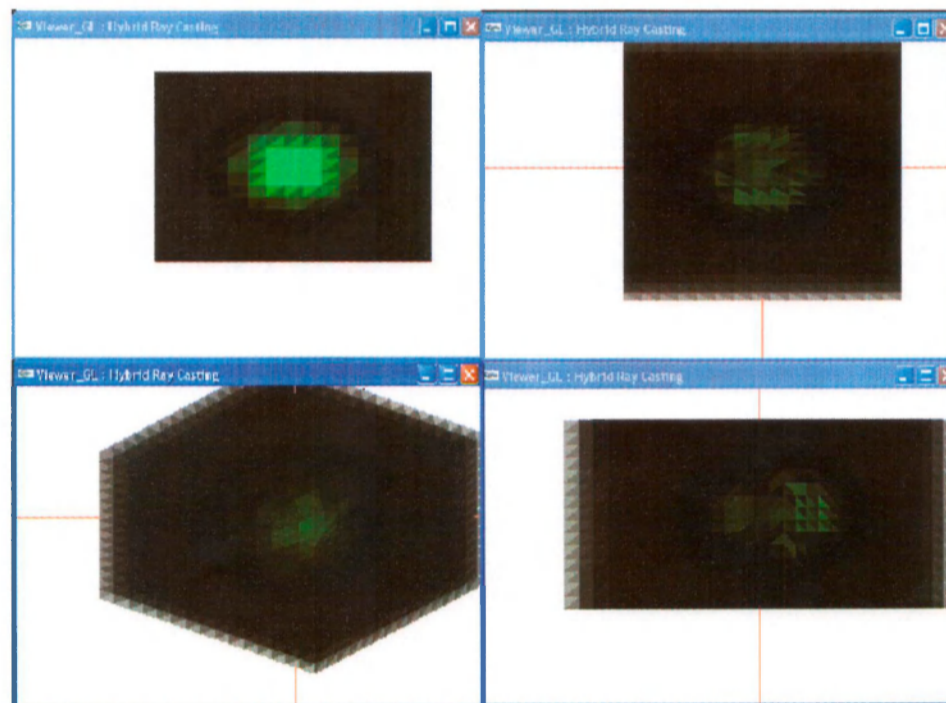
A *figura 5.10* mostra uma série de imagens de um objeto renderizado em um ambiente de teste simples. O volume apresentado contém 20250 células tetraedrais. A imagem obtida é de uma esfera vermelha embutida num cubo semitransparente cinza. As *figuras 5.10(a)*, *5.10(b)* e *5.10(c)* mostram a diferença na visualização quando se utiliza mais que um cone de raios. O cálculo das cores das pré-imagens tomou aproximadamente 12, 25, 40 segundos para os casos a, b, c respectivamente, mas a etapa final de projeção para geração da imagem é feita em tempo real.



(a)



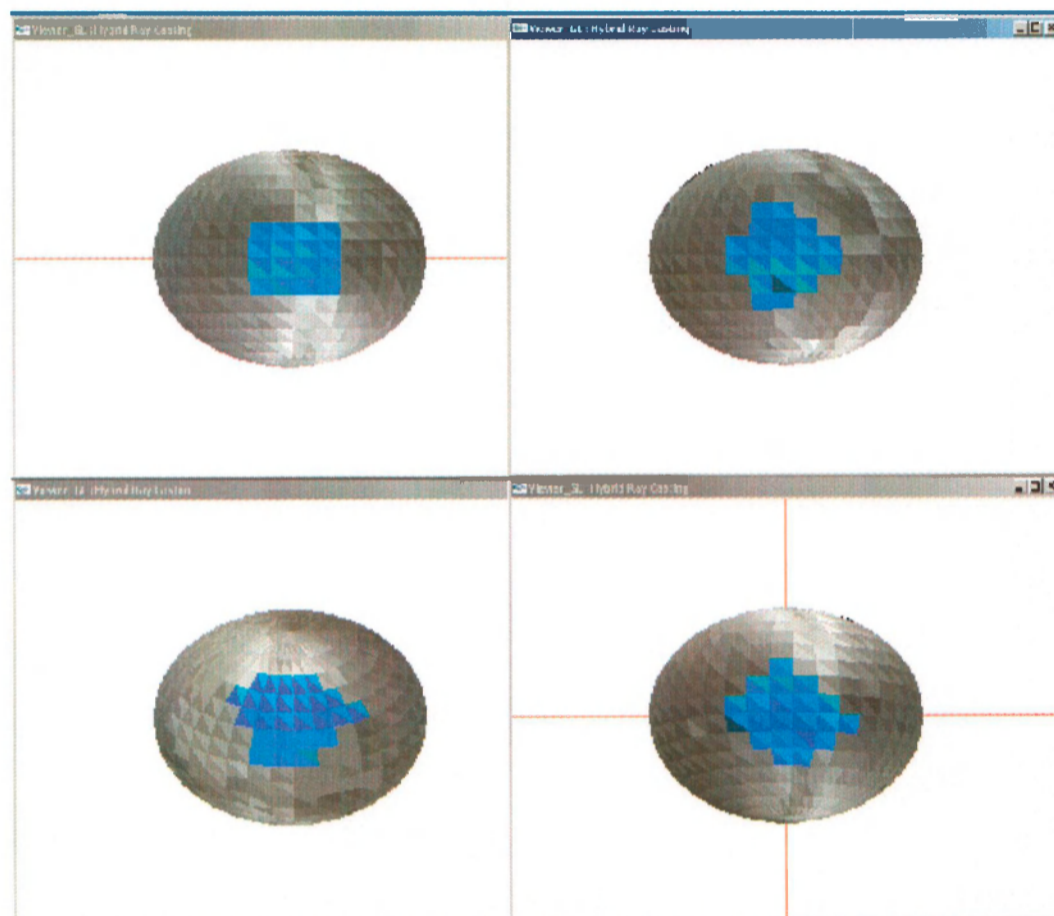
(b)



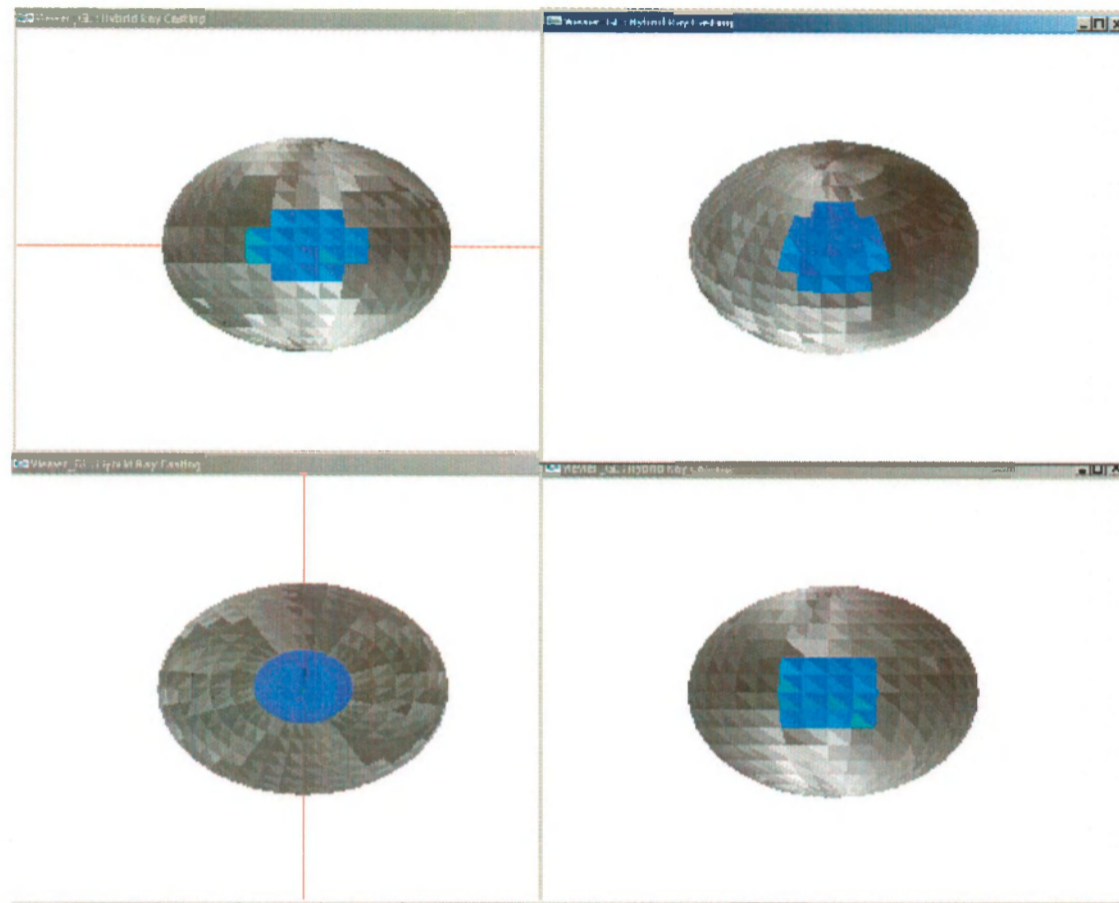
(c)

Figura 5.10. Série de imagens de uma esfera dentro de um cubo: (a) imagens geradas com um cone de quatro direções; (b) imagens geradas com dois cones de quatro direções cada; (c) imagens geradas com três cones de quatro direções cada.

A *figura 5.11* mostra a série de imagens correspondentes a um volume de 34272 células tetraedrais, onde estão representadas uma esfera semitransparente embutida em outras duas esferas semitransparentes de cores distintas. O cálculo das cores das pré-imagens tomou aproximadamente 15, 25, 45 segundos para o caso *5.11a*, *5.11b*, e *5.11c* respectivamente.



(a)



(b)

Figura 5.11. *Serie de imagens de três esferas, cada uma dentro da outra: (a) imagens geradas com um cone de quatro direções; (b) imagens geradas com dois cones de quatro direções.*

5.4. Considerações Finais

Neste capítulo, apresentou-se a base da nova Técnica Híbrida de Visualização para a Exploração de Dados Volumétricos Não Estruturados. O núcleo desta técnica está baseado em uma técnica de *Ray Casting*, a qual denominamos de *Ray Casting direto-superficial*.

Após uma breve introdução, foi apresentado o *pipeline* de visualização da presente técnica híbrida com uma visão global das etapas e processos de desenvolvimento desta técnica. Foram, então, definidas as três etapas principais de visualização. Primeiramente a forma de armazenamento dos dados. Em seguida, foi apresentada a etapa de *rendering* volumétrico baseado no conceito de *Ray Casting*. Finalmente, apresentou-se uma descrição da etapa de *rendering* superficial para realizar a projeção final da cena.

Particularmente, o *Ray Casting* foi escolhido por permitir a visualização de estruturas internas, e pela sua concepção que nos permite projetar uma pré-imagem sobre as superfícies dos objetos da cena, de forma relativamente simples, se comparado com outros métodos. O estudo realizado sobre trabalhos anteriores a respeito deste algoritmo, permitiu um melhor entendimento dos fundamentos nos quais se baseia e permitiu também aproveitar algumas idéias de otimização para promover uma melhor e mais rápida visualização.

CAPITULO 6

Conclusões e Sugestões para Trabalhos Futuros

Conclusões

A técnica de *rendering* híbrida proposta nesse trabalho aproveita as vantagens do *Rendering* Volumétrico Direto para a visualização dos dados internos dos volumes, utilizando para isto transparência; e do *Rendering* de Superfícies para a visualização final, exibindo só as faces superficiais, as que armazenam informação volumétrica.

Esta técnica híbrida opera diretamente sobre dados volumétricos não estruturados, isto é, não realiza reamostragem dos dados em malhas regulares, eliminando desta forma artefatos de conversão produzidos por outros métodos. Além disso, possibilita uma interação entre o usuário e os dados volumétricos, o que não é facilmente obtido com outras técnicas descritas na literatura.

O algoritmo pode ser facilmente paralelizado ao nível do RVD, pois se baseia em uma técnica de lançamento de raios, processo que pode ser feito independentemente, já que os raios são totalmente independentes entre eles.

O RS está baseado no *rendering* das primitivas geométricas que conformam a superfície do volume, o que pode ser feito diretamente em hardware comum.

Como toda técnica de visualização, esta técnica apresenta algumas desvantagens, tais como, que é sensível a variações nos métodos utilizados durante as diferentes etapas de processamento. Mesmo alterações leves podem modificar as características observadas e a qualidade da imagem final.

Dependendo do tipo de aplicação, e do realismo que se deseje, variará diretamente o nível de refinamento da malha utilizada, e o número de raios a serem lançados através do volume desde cada face de bordo.

Considerando os raios lançados, deve-se ter especial cuidado na implementação dos algoritmos de interseção com as células, pois dependendo de como se resolvam os casos especiais de interseção (das faces, arestas ou vértices), artefatos podem aparecer na imagem final. Isto é importante, uma vez que as interseções determinam a amostragem de cor e opacidade, o que repercute no resultado da integração das cores que serão dadas às faces de bordo.

Utilizar conjuntos de dados com malhas refinadas permite uma melhor representação das características internas do volume e atenuam o efeito de serrilhado nas superfícies. Porém, quanto maior o nível de refinamento das malhas, maior é a quantidade de memória consumida e maior o trabalho de processamento necessário.

Sugestões para Trabalhos Futuros

Empregar interpolações, tanto na amostragem feita durante o RVD, quanto durante a iluminação das superfícies durante o RS a fim de suavizar a visualização da imagem final.

Empregar técnicas de multiescala, e níveis de detalhe variáveis junto com o algoritmo híbrido durante a visualização das malhas.

Criar uma versão paralela deste algoritmo que explore arquiteturas distribuídas. Podem ser adotadas mais de uma forma de paralelismo dependendo das características da arquitetura que se tenha a disposição.

Integrar o algoritmo de *rendering* híbrido aos algoritmos de modelagem de objetos para acompanhar os resultados da modelagem no espaço 3D.

Desenvolver ferramentas que permitam identificar automaticamente estruturas internas do volume como apoio ao análise dos dados. Isto permitiria focar o estudo em determinadas regiões dos dados, guiado pela visualização das regiões de interesse.

REFERÊNCIAS BIBLIOGRÁFICAS

- [AG89] G. S. Almasi and A. Gottlieb, "Highly parallel computing", The Benjamin/Cummings Publishing Company, Inc., 1989.
- [BCL99] C. L. Bajaj, E. J. Coyle and K. Lin. "Tetrahedral meshes from planar cross-sections". *Comput. Methods Appl. Mech. Eng.*, vol.179, pp.31-52, 1999.
- [BCL96] C. L. Bajaj, E.J. Coyle and K. Lin. "Arbitrary Topology Shape Reconstruction from Planar Cross Sections". *Graphical Models and Image Processing*, vol.58, no.6, pp. 524-543, 1996.
- [Ba00] M. Baker, "3D Theory: Spatial Decomposition," Available at <http://www.martinb.com/threed/solidmodel/spatialdecomposition>
- [BCEG91] K. W. Brodie, L. A. Carpenter, R. A. Earnshaw, J. R. Gallop, R. J. Hubbard, D. D. Munford, C. D. Osland, and P. Quarendon (eds.), "Scientific Visualization Techniques and Applications", Springer Verlag Press, 1991.
- [Bo88] J.D.Boissonnat. "Shape Reconstruction from Planar Cross Sections". *Computer vision, Graphics and image Processing* vol.44, pp.1-29, 1988.
- [BKS97] P. Bunyk, A E. Kaufman, and C. T. Silva, "Simple, Fast, and Robust Ray Casting of Irregular Grids". New York, USA, 1997.
- [CV00] M.M.Carneiro and L.Velho, "Um estudo de algoritmos para visualização simultanea de dados volumétricos e superfícies poligonais", PUC-Rio/ing.MCC 14/00, março,2000.
- [Ch93] J. S. Challinger, "Scalable Parallel Direct Volume Rendering for Non Rectilinear Computational Grids". University of California, California, USA, 1993.
- [Ch92] J. S. Challinger. "Parallel Volume Rendering for Curvilinear Volumes," *Proceedings of the Scalable High Performance Computing Conference*, IEEE Computer Society Press, pp.14-21, Apr. 1992.
- [CTT00] Chuan-kai Yang, Tulika Mitra, Tzi-cker Chiueh, "On-the-Fly Rendering of Losslessly Compressed Irregular Volume Data", in *IEEE Visualization '2000*, Salt Lake City, Utah, October 2000.
- [CMS97] P. Cignoni, C. Montani, and R. Scopigno, "Tetrahedral Based Volume Visualization," *Instituto CNUCE - C.N.R.*, Pisa, Italy. 1997.
- [CMPS97] P. Cignoni, C. Montani, E. Puppo, and R. Scopigno, "Speeding up iso-surface extraction using interval trees," *IEEE Trans. On Visualization and Computer Graphics*, vol.3, no.2, 1997.
- [CLLC88] H. E. Cline, W. E. Lorensen, S. Ludke, D. R. Crawford, and B. C. Teeter, "Two Algorithms for Three-dimensional Reconstruction of Tomograms," *Medical Physics*, vol.15, no.3, pp.320-327, Jun. 1988.
- [CKMM99] J. Comba, J. Klosowski, N. Max, J. Mitchell, C. Silva, and P. Williams, "Fast Polyhedral Cell Sorting for Interactive Rendering of Unstructured Grids", *EUROGRAPHICS'99*, vol.18, no.3, 1999.
- [DCH88] R. A. Drebin, L. Carpenter and P. Hanrahan, "Volume Rendering", *Computer Graphics*, vol.22, no.4, pp. 65-74, Aug. 1988.
- [EKMM91] J. Ellis, G. Kedem, R. Marisa, J. Menon, and H. Voelcker, "Breaking barriers in solid modeling", *mechanical engineering*, vol. 113, no. 2, pp. 28-34, bebruary, 1991.

- [El92] T. Elvins, "A Survey of Algorithms for Volume Visualization", ACM SIGGRAPH, 3-3.14 (course notes 1), 1992.
- [EPO91] A. B. Ekoule, F. C. Peyrin, and C. L. Odet, "A Triangulation Algorithm from Arbitrary Shaped Multiple Planar Contours," ACM Transaction on Graphics, vol.10, no.2, pp.182-199, Apr. 1991.
- [Fr91] M. Fruhauf, "Combining Volume Rendering with Line and Surface Rendering", EUROGRAPHICS'91, 1991.
- [Fr94] T. Fruhauf, "Raycasting of Nonregularly Structured Volume Data," Computer Graphics Forum (EUROGRAPHICS'94), vol. 13, no. 3, 1994.
- [FKU97] H. Fuchs, Z. M. Kedem, and S. P. Uzelton, "Optimal Surface Reconstruction from Planar Contours," Communication of the ACM, vol.20, no.10, pp.693-702, Oct. 1997.
- [FKN80] H. Fuchs, Z. M. Kedem, and B. Naylor, "On Visible Surface Generation by A Priori Tree Structures," Computer Graphics, vol. 14, no. 3, pp. 124-133, 1980. Proc. SIGGRAPH '80.
- [Ga90] M. P. Garrity, "Raytracing Irregular Volume Data," Computer Graphics (San Diego Workshop on Volume Visualization), vol. 24, pp. 35-40, Nov. 1990.
- [Gi90] C. Giertsen, "Volume Visualization of Sparse Irregular Meshes," Computer Graphics, vol.12, no.2, pp.185-192, Jul. 1989.
- [GLDH00] M. H. Gross, L. Lippert, R. Dittich, and S. Häring, "Two Methods for Wavelet-Based Volume Rendering", Institute for Information Systems Computer Graphics Research Group, Swiss Federal Institute of Technology Zurich, Internal Report # 247, 2000.
- [GC91] D. Gordon, and S. Chen, "Front to Back Display of BSP Trees", IEEE Computer Graphics and Applications, pp.79-82, Sep. 1991.
- [Go71] H. Gouraud, "Continuous Shading of Curved Surfaces", IEEE Transactions of Computers, vol.20,no.6,pp.623-629,1971.
- [Gu88] J. L. Gustafson, "Reevaluating Amdahl's Law", Communications of the ACM, vol31, no. 5, pp. 532-533, may, 1988.
- [HM90] D. Helmbold and C. MacDowell, "Modelling speedup(n) greater than n", IEEE Transactions on Parallel and Distributed Systems, vol.1, no.2, pp.250-256, april,1990.
- [HL79] G. T. Herman, and H. K. Liu, "Three-dimensional display of Human Organs from Computed Tomograms," Computer Graphics and Image Processing, vol.9, no.1, pp.1-21, Jan. 1979.
- [Ho93] R. M. Hord, "Parallel supercomputing in MIMD architectures", CRC press. Inc.,1993.
- [Ja88] D. Jackel, "Reconstructing Solids from Tomographic Scans - The PARCUM II System," Advances in Computer Graphics Hardware II, pp. 101-109, Springer International, 1988.
- [JC94] M. W. Jones and M. Chen. "A New Approach to the Construction of Surfaces from Contour Data". Proc. EUROGRAPHICS94, vol.13, no.3, pp. 75-84, 1994.
- [Ka98] A Kaufman, "Advances in Volume Visualization", SIGGRAPH'98, Course Notes no.24, Orlando, 1998.
- [Ka94] A. E. Kaufman, et. al., "Research Issues in Volume Visualization," IEEE Computer Graphics and Applications, vol. 14, no. 2, pp. 63-67, Mar. 1994.
- [Ka91] A. Kaufman, "Introduction to Volume Visualization," Volume Visualization, A.Kaufman (ed.), IEEE Computer Society Press, pp. 1-18, 1991.

- [Ka97] A Kaufman, "Volume Visualization: Principles and Advances", State University of New York, Stony Brook, NY, USA, 1997.
- [KB88] A Kaufman, and R. Bakalash, "Memory and Processor Architecture for 3D Voxel-Based Imagery," IEEE Computer Graphics and Applications, vol. 8, no.11, pp.10-23, Nov. 1988.
- [KBCY90] A. Kaufman, R. Bakalash, D. Cohen, and R. Yagel, "A survey of architectures for volume rendering", IEEE engineering in Medicine and Biology Magazine, vol.9, no.4, pp.18-23,1990.
- [KS86] A. Kaufman, and E. Simony "Scan-conversion Algorithm for Voxel-base Graphics," Proceeding, ACM Workshop on Interactive 3D Graphics, Chapel Hill, NC, pp. 45-75, Oct. 1986.
- [KYC90] A. Kaufman, R. Yagel, D. Cohen, "Intermixing Surface and Volume Rendering, 3D Imaging in Medicine: Algorithms, Systems and Applications", Springer Verlag, pp. 217-228,1990.
- [Ke97] E. Kepler, "Approximating Complex Surfaces by Triangulation of Contour Lines," IBM Journal of Research and Development, vol.19, no.1, pp.2-11, Jan. 1997.
- [KSS00] R. Klein, A. Schilling, and W. Straber. "Reconstruction and Simplification of Surfaces from Contours". Graphical Models, vol.62, pp.429-443, 2000.
- [Ko90] K. Koyamada, "Volume Visualization for the Unstructured Data," SPIE vol.1259 Extracting Meaning from Complex Data: Processing, Display, interaction, 1990.
- [Ko92] K. Koyamada, "Fast Traversal of Irregular Volumes." Visual Computing -- Integrating Computer Graphics and Computer Vision, pp. 295-312. Springer Verlag, 1992.
- [La95] P. Lacroute. "Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation," Ph.D. dissertation, Technical Report CSI-TR-95-678, Stanford University, 1995. Available at <http://graphics.stanford.edu/papers/theses.html>
- [LL94] P. Lacroute and M. Levoy, "Fast Volume Rendering using a Shear-Warp Factorization of the Viewing Transformation", Computer Graphics, vol.28, no.3, pp.451-458, Jul. 1994.
- [LH91] D. Laur and P. Hanrahan, "Hierarchical Splatting: A Progressive Refinement Algorithm for Volume Rendering", Computer Graphics, vol.25, no.4, pp.285-288, Jul. 1991.
- [Le88] M. Levoy, "Volume Rendering: Display of Surfaces from Volume Data", IEEE Computer Graphics and Applications, pp. 29-37, May. 1988.
- [Le90a] M. Levoy, "A Taxonomy of Volume Visualization Algorithms", ACM SIGGRAPH, pp.8-12 (course notes 11), 1990.
- [Le90b] M. Levoy, "Volume Rendering by Adaptive Refinement," The Visual Computer, vol.6, no.1, pp.2-7, Feb. 1990.
- [Le90c] M. Levoy, "A Hybrid Ray Tracer for Rendering Polygon and Volume Data", IEEE Computer Graphics and Applications, vol.10, no.3, pp. 33-40, Mar 1990.
- [Le90d] M. Levoy, "Ray Tracing of Volume Data", SIGGRAPH'90 course notes, Volume Visualization Algorithms and Architectures, August,1990,pp.120-147.
- [Le90e] M. Levoy., "Efficient Ray Tracing of Volume Data", Computer Graphics, Proceedings of SIGGRAPH '90, pp 157-167,1990.
- [LFPR90] M Levoy, H. Fuchs, S. M. Pizer, J. Rosenman, E. I. Shaney, G. W. Sherouse, V. Interrante, and J. Kiel "Volume Rendering in Radiation Treatment Planning", Proceedings of the first Conference on Visualization in Biomedical Computing, IEEE Computer Society Press, Atlanta, Georgia, May, 1990, pp. 4-10.

- [LW90] M. Levoy and R. Whitaker, "Gaze-directed Volume Rendering", *Computer Graphics*, vol.24, no.2, pp.217-223, 1990. Proceedings of 1990 symposium on interactive 3D graphics.
- [LGK02] L. Lippert, M. H. Gross, C. Kurmann, "Compression Domain Volume Rendering for Distributed Environments", Department of Computer Science, Federal Institute of Technology, Zürich, Switzerland, Internal Report #263, 2002.
- [LC87] W. E. Lorensen and H.E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm, *Computer Graphics (Proc. SIGGRAPH)*, pp.163-169, Jul. 1987.
- [Ma95] K. L. Ma, "Parallel Volume Rendering for Unstructured-Grid Data on Distributed Memory Machines", *Proc. IEEE/ACM Parallel Rendering Symposium '95*, pp. 23-30. 1995.
- [Ma93] T. Malzbender, "Fourier Volume Rendering", *ACM Transactions on graphics*, vol.12, no.3, pp.233-250, 1993.
- [MHK95] X. Mao, L. Hong, and A. Kaufman, "Splating of Curvilinear Grids", *IEEE Visualization '95*, pp. 61-68, 1995.
- [Max95] N. Max, "Optical Models for Direct Volume Rendering", *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, no. 2, pp. 99-108, Jun. 1995.
- [MHC90] N. Max, P. Hanrahan, and R. Crawfis, "Area and Volume Coherence for Efficient Visualization of 3D Scalar Functions", *Computer Graphics (San Diego Workshop on Volume Visualization)*, vol. 24, pp. 27-33, Nov. 1990.
- [MDB87] B. McCormick, T. De Fanti, and M. Brown, "Visualization in Scientific Computing", *Computer Graphics*, vol. 21, no.6, Nov. 1987.
- [Me82] D. J. Meagher, "Applying Solids Processing Methods to Medical Planning", in proceedings of the IEEE Computer Society Conference on Pattern Recognition and Image Processing, Jun., 1982.
- [MMS94] J. S. B. Mitchell, D. M. Mount, and S. Suri, "Query-Sensitive Ray Shooting", *Proc. 10th Annual ACM Symposium of Computing Geometry*, pp. 359-368, Jun. 1994.
- [MK92] T. Miyazawa and K. Koyamada, "A high-speed integrated renderer for Interpreting Multiple 3D volume Data", *The Journal of Visualization and Computer Animation*, 3,65-83, 1992.
- [NH91] G. M. Nielson and B. Hanrmann, "The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes", *Proceedings of the IEEE Visualization '91 Conference*, IEEE Society Press, pp. 51-58, Oct. 1991.
- [NL92] J.Nieh and M.Levoy, "Volume Rendering on Scalable Shared-Memory MIMD Architecture", *Workshop on Volume Visualization '92*, ACM, pp. 17-24, 1992.
- [NMS00] L.G Nonato, R. Minghim, and M.H. Shimabukuro. "Qualitative Analysis of two Reconstruction Techniques and their application in Dentistry". *Journal of Electronic Imaging*, pp. 385-393, 2000
- [NSG90] K.L. Novins, F. Sillion, and D.P. Greenberg, "An Efficient method for volume rendering using perspective projection", *Computer Graphics*, vol.24,no.5,pp 95-102, 1990.
- [OUT85] T. Ohashi, T. Uchiki, and M. Tokoro, "A three-dimensional shaded display method for voxel-based representations", in *Proceedings of EUROGRAPHICS 1985*, pp.221-232, National computer graphics association, September, 1985.
- [PSG99] A.C. Paiva, R. B. Seixas, and M. Gattas, "Introdução a Visualização Científica", monografia em Ciências da Computação, no.3/99,Departamento de Informatica, PUC-Rio,1999.

- [Pa97] J. Pawasauskas, "Volume Visualization with Ray Casting", CS563 – Advanced Topics in Computer Graphics, Feb. 1997.
- [Ph97] D. Phillips, "Visualizing Volumes", Computer Graphics World, Jul. 1997.
Available at <http://cgw.pennnet.com/home.cfm>
- [PT75] Phong, Bui Tuong, "Illumination for Computer Generated Pictures", Communications of the ACM, vol.18, no.6, pp.311-317, Jul. 1975.
- [Pr96] C. E. Prakash, "Parallel Voxelization Algorithms for Volume Rendering of Unstructured Grids", Ph.D. thesis, Supercomputer Centre, Indian Institute of Science, 1996.
- [RW92] S. Ramamoorthy and J. Wilhelms, "An Analysis of Approaches to Ray-Tracing Curvilinear Grids", Tech Report UCSC-CRL-92-07, U. of California, Santa Cruz, 1992.
- [RCH99] R. A. Robb, J. J. Camp, and D. P. Hanson, "Computer aided surgery and treatment planning at the Mayo Clinic", Biomedical imaging resource, Mayo Foundation, Rochester, MN, 1999.
- [SFLP00] R. Samanta, T. Funkhouser, K. Li, and J. Pai Singh, "Hybrid Sort-First and Sort-Last Parallel Rendering with a Cluster of PCs", Princeton University, 2000.
- [SGC99] A.E. Schmidh, M. Gattass, P.C. Carvalho, "Combined 3D Visualization of Volume Data and Polygonal Surfaces Using a Shear-Warp Algorithm", Monografia em Ciências da Computação, no.5, Departamento de Informatica, PUC-Rio, 1999.
- [SS92] P. Schöder and G. Stoll, "Data Parallel Volume Rendering as Line Drawing", Workshop on Volume Visualization '92, ACM, pp. 25-32, 1992.
- [SS91] P. Schöder and B. J. Salem, "Fast Rotation of Volume Data on Data Parallel Architecture", Course Notes 8: State of the Art in Volume Visualization, ACM SIGGRAPH'91 Conference, 1991.
- [Sc99a] G. Scott Owen, "Ray Tracing", 1999.
Available at <http://www.siggraph.org/education/materials/HyperGraph/raytrace/rtrace0.htm>
- [Sc99b] G. Scott Owen, "Surface Fitting Algorithms", 1999.
Available at <http://siggraph.org/education/materials/HyperVis/vistech/volume/surface.htm>
- [SGDM95] R. De B. Seixas, M. Gattass, L. H. De Figueiredo, L. F. Martha, "Visualização Volumétrica com Otimizações de Ray Casting e Detecção de Bordas", Anais do VIII SIBGRAP, pp.281-286, Oct. 1995.
- [ST90] P. Shirley and A. Tuchman, "A Polygonal Approximation to Direct Scalar Volume Rendering", Computer Graphics, vol. 24, no.5, pp.63-70, Nov. 1990.
- [SM97] C. Silva and J. Mitchell, "The Lazy Sweep Ray Casting Algorithm for Rendering Irregular Grids", IEEE Transactions on Visualization and Computer Graphics, vol. 3, no. 2, Jun. 1997.
- [Si95] C. Silva, "Parallel Processing for Volume Visualization", Dept of Computer Science SUNY at Stony Brook, NY, 1995. Available at <http://www.cs.sunysb.edu/~csilva/papers/rpe>
- [SK95] C. Silva and A Kaufman, "Parallel Performance Measures for Volume Ray Casting", Dept of Computer Science, SUNY at Stony Brook, NY, 1995.
- [SBM94] C. Stein, B. Becker, and N. Max, "Sorting and Hardware Assisted Rendering for Volume Visualization", Symposium on Volume Visualization, pp. 83-90, October 1994.
- [TPN93] D. Tost, A. Puig, and I. Navazo, "Visualization of mixed scenes based on volume and surfaces". Fourth Eurographics workshop on Rendering, pp.281-293, 1993.

- [TL88] T. Totsuka and M. Levoy, "Frequency Domain Volume Rendering", *Computer Graphics (Proc. SIGGRAPH)*, pp.59-64, 1988.
- [TT84] H. K. Tuy, L. T. Tuy, "Direct 2-D Display of 3-D Objects", *IEEE Computer Graphics and Applications*, vol.4, no.10, pp.29-33, Nov. 1984.
- [TCT97] Tzi-cker Chiuch, Chuan-kai Yang, Taosong He, Hanspeter Pfister, and Arie Kaufman, "Integrated Volume Compression and Visualization", *IEEE Visualization '97*, Phoenix, AZ, October 1997.
- [UK88] C. Upson and M. Keeler, "V-Buffer: Visible Volume Rendering", *Computer Graphics*, vol.22, no.4, pp.59-64, Aug. 1988.
- [Us91] S. Uzelton, "Volume Rendering for Computational Fluid Dynamics: Initial Results", Tech Report RNR-91-026, Nasa Ames Research Center, 1991.
- [VW93] A. Van Gelder and J. Wilhelms, "Rapid exploration of curvilinear grids using direct volume rendering", *IEEE Visualization '93*, pp.70-77, 1993.
- [VH92] M. Vetterli and C. Herley, "Wavelets and Alter banks: Theory and design", *IEEE Transactions on Signal Processing*, vol. 40, no. 9, pp. 2207-2232, 1992.
- [WR00] M. A. Westenberg and J. B. T. M. Roerdink, "Frequency Domain Volume Rendering by the Wavelet X-ray Transform", *IEEE Transactions on Image Processing*, vol. 9, no. 7, Jul. 2000.
- [We94] R. Westermann, "A Multiresolution Framework for Volume Rendering". *Symp. On Volume Visualization*, Washington, pp. 51-58, 1994.
- [We90] L. Westover, "Footprint Evaluation for Volume rendering", *Computer Graphics*, vol.24, no.4, pp.367-376, Aug. 1990.
- [Wh92] S. Whitman, "Multiprocessor methods for computer graphics rendering", Jones and Bartlett, 1992.
- [WCAR90] J. Wilhelms, J. Challinger, N. Alper, S. Ramamoorthy, and A. Vaziri, "Direct Volume Rendering of Curvilinear Volumes", *Computer Graphics (San Diego Workshop on Volume Visualization)*, vol. 24, pp. 41-47, Nov. 1990.
- [Wi93] J. Wilhelms, "Pursuing Interactive Visualization of Irregular Grids", *Visual Computer*, vol. 9, no. 8, 1993.
- [Wi91] J. Wilhelms, "Decisions in Volume Rendering", *State of the Art in Volume Visualization*, ACM SIGGRAPH '91 Course Notes, Course Number 8, ACM Press, Aug. 1991, 1.1-1.11.
- [Wi92] P. Williams, "Visibility Ordering Meshed Polyhedra", *ACM Transactions on Graphics*, vol. 11, no. 2, 1992.
- [WMS98] P. L. Williams, N. L. Max, and C. M. Stein, "A High Accuracy Volume Renderer for Unstructured Data", *IEEE Transactions on Visualization and Computer Graphics*, vol. 4, no. 1, Jan.-Mar. 1998.
- [YCK92] R. Yagel, D. Cohen, and A. Kaufman, "Discrete Ray Tracing", *IEEE Comp. Graphics & Applications*, vol.12, no.5, pp.19-28, 1992.
- [YK92] R. Yagel and A. Kaufman, "Template-Based Volume Viewing", *IEEE Computer Graphics and Applications*, vol.12, no.5, pp.19-28, Sep. 1992.
- [YRLS96] R. Yagel, D. Reed, A. Law, P-W. Shih, and N. Shareef, "Hardware Assisted Volume Rendering of Unstructured Grids by Incremental Slicing", *IEEE-ACM Volume Visualization Symposium*, pp. 55-62, Nov. 1996.

- [ZS99] M. N. Zakaria and M. Y. Saman, "Hybrid Shear-Warp Rendering", Proceedings of the ICVC, Goa, India, 1999.
- [ZL94] M. Zuffo and R. Lopes, "A High Performance Direct Volume Rendering Pipeline", Anais do VII SIBGRAPI , pp. 241–248, 1994.