Um Estudo de Algoritmos para Visualização Simultânea de Dados Volumétricos e Superfícies Poligonais

Marcelo Medeiros Carneiro
PUC-Rio/Departamento de Informática
mmc@inf.puc-rio.br

Luiz Velho
IMPA - Instituto de Matemática Pura e Aplicada
lvelho@visgraf.impa.br

PUC-RioInf.MCC14/00 Março/2000

ABSTRACT: Some scientific visualization applications require rendering not only of volume data but also of polygonal surfaces in the same scene. This work presents some basic direct volume rendering algorithms in addition to some changes in them so as to allow rendering of both representations at the same time. Hybrid Ray Casting and Shear-Warp algorithms are discussed and the main problems related to the integration of volume and surfaces are presented. This study allow us evaluate computational cost and image quality in these hybrid algorithms.

KEYWORDS: Volume Rendering, Ray Casting, Shear-Warp, Splatting, Hybrid Volume Rendering.

RESUMO: Algumas aplicações em visualização científica requerem não apenas a renderização de dados volumétricos, mas também de superfícies poligonais em uma mesma cena. Este trabalho apresenta os principais algoritmos de visualização volumétrica direta e algumas variações de forma a permitir a renderização simultânea de dados volumétricos e superfícies poligonais. São apresentadas versões de algoritmos híbridos de *Ray Casting* e *Shear-Warp* e as principais dificuldades envolvidas nessa integração. Os resultados obtidos permitem uma avaliação do custo computacional e da qualidade final das imagens obtidas.

PALAVRAS-CHAVE: Visualização Volumétrica, *Ray Casting*, *Shear-Warp*, *Splatting*, Renderização Volumétrica Híbrida.

Sumário

Lista de Figuras

1 INTRODUÇÃO

Visualização Volumétrica é um termo geral utilizado para descrever técnicas que permitem a projeção de um conjunto de dados no espaço tridimensional (conhecido também por dados volumétricos) em uma superfície de visualização bidimensional [Elvis, 92]. Essas técnicas auxiliam o entendimento de propriedades e estruturas contidas dentro do volume, pois permitem a visualização do dado como um todo.

Em sua grande maioria, os algoritmos de visualização volumétrica estão muito mais preocupados em resolver o problema da visualização, não dando maior importância ao problema da modelagem [Tosti e outros, 93]. Isso acontece porque o dado volumétrico é tratado apenas como um conjunto de grandezas escalares ou vetoriais no espaço.

Quando o problema da modelagem do dado volumétrico é levado em consideração, os algoritmos podem tirar proveito da estrutura do dado volumétrico, pois, compreendendo melhor seu conteúdo, podem ganhar uma certa inteligência. Com isso, mecanismos mais eficazes de navegação e visualização podem ser implementados. Para exemplificar, pode-se citar o uso de vínculos para auxiliar a navegação do usuário durante um exame virtual (por exemplo, endoscopia assistida por computador). Durante o exame, a navegação fica restrita apenas aos locais permitidos (por exemplo, no interior de uma artéria), impedindo que o usuário alcance uma região fora da área de interesse.

Em muitas aplicações, é necessário não apenas visualizar os dados que formam o volume, mas também algum outro tipo de representação, por exemplo, superfícies ou outras primitivas geométricas. Por isso, torna-se bastante útil o desenvolvimento de algoritmos híbridos que permitem a visualização simultânea de dados volumétricos e superfícies poligonais, por exemplo.

Muitas aplicações na área médica requerem a utilização de algoritmos híbridos de visualização volumétrica [Robb, 99]. Entre elas, pode-se citar desenvolvimento de implantes e próteses, conforme mostrado na Figura 1.1. A imagem foi obtida através de um exame de tomografia computadorizada (CT). Pela figura, pode-se notar que as dimensões da prótese foram calculadas através do espelhamento do rosto do paciente. Esse tipo de aplicação requer a utilização de modelos geométricos para serem devidamente encaixados no volume.

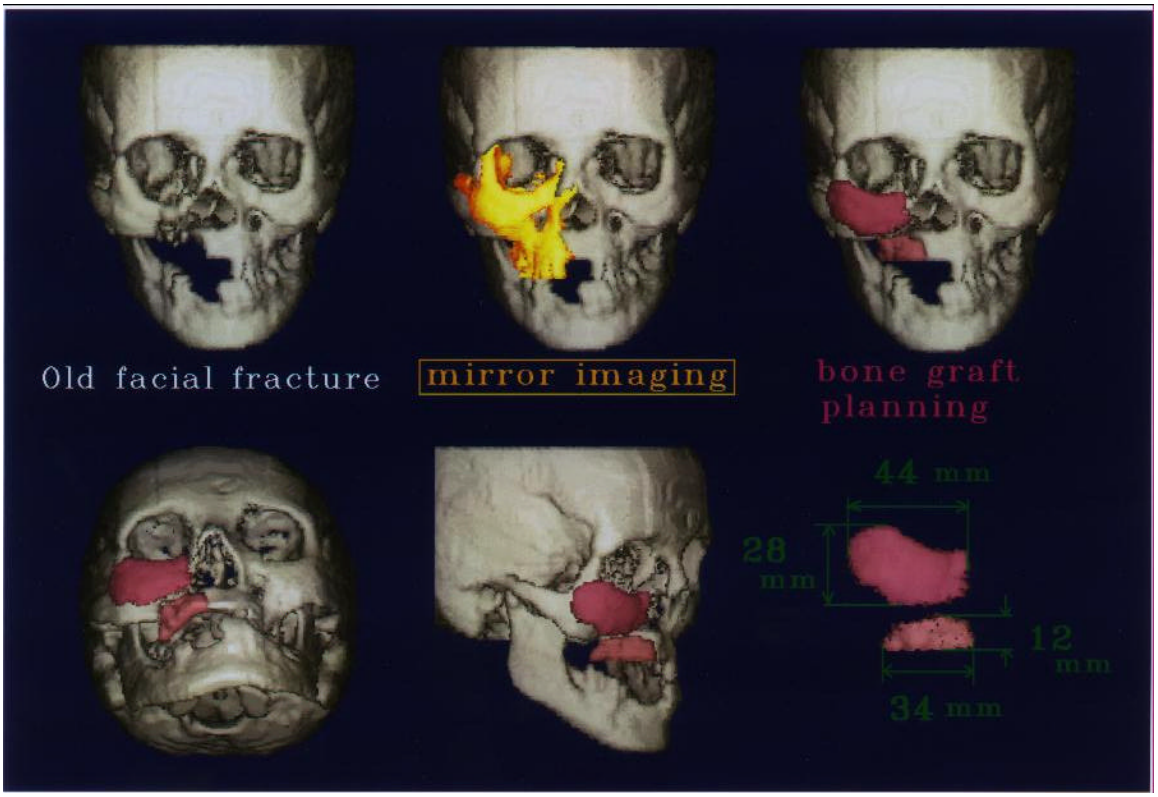


Figura 1.1: Desenvolvimento de uma prótese óssea [Robb, 99].

Durante o planejamento de uma radioterapia (RTP, *Radiation Therapy Planning*) também pode ser importante a utilização de modelos geométricos sobre o volume. Alguns tipos de tumores não podem ser retirados através de intervenção cirúrgica, porém uma dose suficiente de radiação pode ser capaz de destruí-los. Entretanto, essa radiação pode também atingir outras partes do corpo, destruindo células sadias. O objetivo de um tratamento de radioterapia é, portanto, eliminar somente os tumores, preservando as demais estruturas do organismo (por exemplo, olhos e nervo ótico). Isso requer a utilização de ferramentas adequadas que permitam ao médico simular o tratamento e corrigir a posição e intensidade da radiação, de forma a não prejudicar o paciente. A Figura 1.2 ilustra esse procedimento. A imagem foi obtida através de um exame de ressonância magnética (MRI). As áreas coloridas na parte superior da figura indicam o nível de radiação (vermelho: 100% de radiação, azul: 30%). A imagem inferior esquerda mostram as estruturas sensíveis a radiação (azul: olhos, amarelo: nervo ótico, verde: raiz do cérebro, vermelho: tumor). A imagem à direita mostra como a radiação atinge as áreas

sensíveis, onde pode-se notar que o tratamento deve ser corrigido, pois uma parte da raiz do cérebro foi atingida.

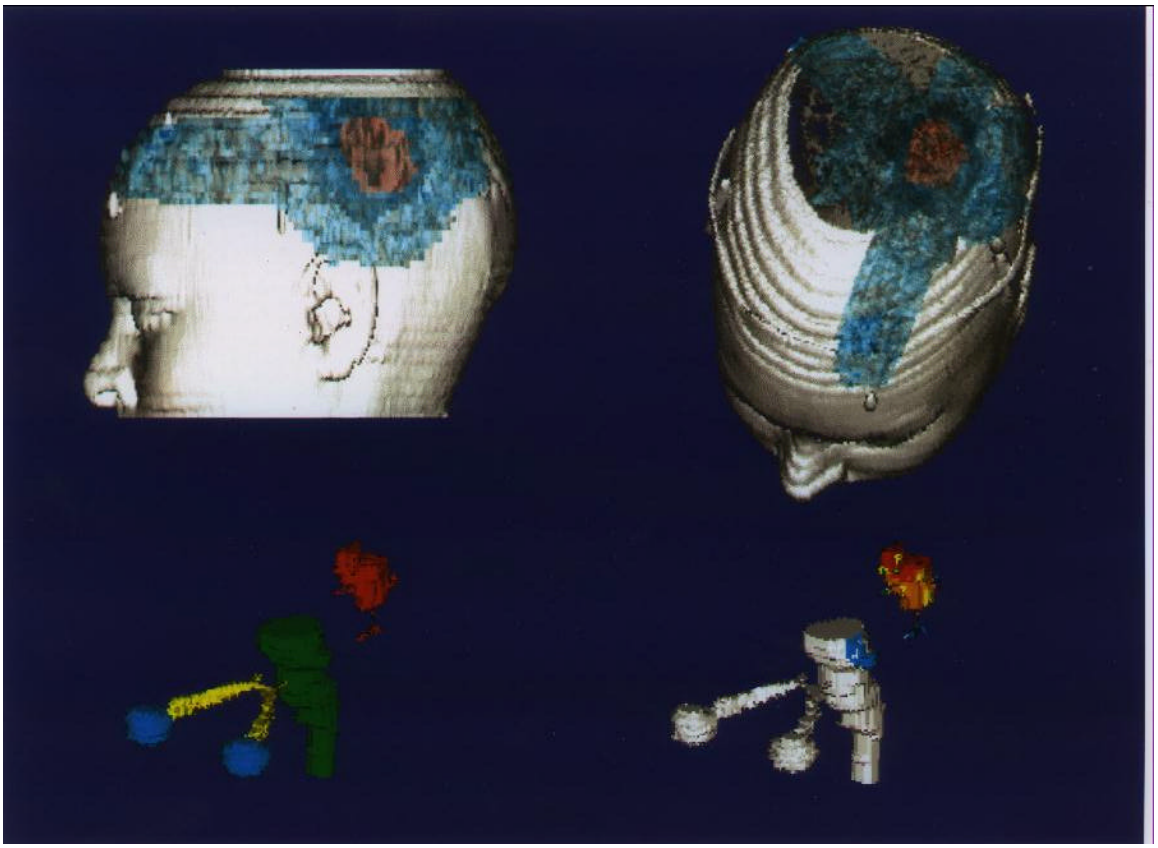


Figura 1.2: Planejamento de um radioterapia [Robb, 99].

A Visualização Volumétrica ainda é um campo da computação gráfica com grandes possibilidades de crescimento. Muitos algoritmos foram desenvolvidos e alguns aperfeiçoados visando melhorar a qualidade das imagens ou diminuir seu tempo de geração, permitindo o desenvolvimento de sistemas interativos. Pode-se citar muitas aplicações de técnicas de visualização volumétrica, porém uma das mais importantes e relevantes ainda é na medicina (imagens médicas). Nesse campo, o dado volumétrico é usualmente obtido através de algum processo de aquisição, seja por tomografia computadorizada (CT), ressonância magnética (MRI), microscopia confocal, entre outros [Paiva e outros, 99].

Tipicamente, em uma aplicação em imagens médicas, o dado volumétrico é apresentado como uma seqüência de fatias bidimensionais (*slices*, normalmente com resolução de 512x512 pontos), separadas por uma pequena distância (poucos milímetros).

Cada fatia representa um seção transversal de um objeto de estudo (um órgão do corpo humano, por exemplo) e o conjunto das fatias permite a construção de uma imagem tridimensional do objeto, auxiliando o médico no diagnóstico de uma doença, tratamento mais adequado ou planejamento de uma cirurgia.

Além de larga aplicação na medicina, técnicas de visualização volumétrica também são muito empregadas em outras áreas da ciência, por exemplo, meteorologia, geologia, engenharia mecânica etc. Entre essas áreas, destaca-se a utililização de visualização volumétrica na indentificação de estruturas geológicas sob a superfície do solo. Isso permite, através da interpretação de dados sísmicos, inferir propriedades do terreno, auxiliando o geólogo na identificação de bacias de petróleo [Gerhardt e outros, 98].

O principal objetivo deste trabalho é apresentar um estudo de algoritmos híbridos de visualização científica, que são capazes de tratar não apenas de dados volumétricos, mas também de objetos geométricos, tais como superfícies poligonais. Para isto, primeiro são apresentadas as principais características do dado volumétrico, isto é, de que forma ele pode ser representado e manipulado. A seguir, são apresentadas algumas técnicas que permitem a conversão de uma representação volumétrica para uma representação poligonal e vice versa. Depois os principais algoritmos de visualização volumétrica são resumidamente apresentados e, a seguir, são estudados algumas modificações nesses algoritmos de forma a torná-los capazes de visualizar cenas compostas por diferentes representações.

2 REPRESENTAÇÃO DO VOLUME

Genericamente, o dado volumétrico pode ser definido por uma função `f` do espaço tridimensional:

$$f: \Re^3 \rightarrow \Re^n$$

Essa função pode ser analítica ou procedural mas, na maioria dos casos, `f` é uma função discreta. Isso significa que a propriedade volumétrica é definida apenas em alguns pontos do espaço. Entretanto, durante a visualização, é necessário que o dado volumétrico seja reconstruído, portanto é preciso que f seja contínua. Isso requer a utilização de uma função de interpolação.

A forma mais simples de interpolação, conhecida por interpolação de ordem zero, simplesmente associa o valor da função `f` em qualquer ponto do espaço como sendo o valor da função no ponto mais próximo onde ela é definida. Esse método de interpolação gera um conjunto de regiões com valores constantes de `f` ao redor dos pontos onde a função é definida. Cada uma dessas regiões, onde a propriedade do dado volumétrico não varia, é chamada de *voxel* (*volume element*, uma analogia ao caso 2D, *pixel*).

Uma forma mais apropriada de interpolação, conhecida por interpolação de primeira ordem (ou interpolação trilinear), assume que a propriedade volumétrica varia linearmente em cada uma das três direções axiais. Isso permite a geração de imagens de melhor qualidade, pois a interpolação é mais suave. Nesse caso, o conceito de *voxel* deixa de existir. Entretanto, o dado volumétrico pode ser visto como sendo formado por um conjunto de células adjacentes e fortemente unidas, onde o valor da propriedade volumétrica é definida em cada vértice da célula (tipicamente oito vértices) e interpolada no interior da mesma. Outras formas de interpolação de ordem mais elevadas também podem ser utilizadas. Apesar de muitos autores tratarem dessa forma, ainda não se pode dizer que há um concenso na literatura com relação ao conceito de *voxels* e células.

O dado volumétrico pode ser obtido através de simulação, onde um modelo matemático é utilizado para gerar o volume (por exemplo, um simulador de reservatório de petróleo), ou aquisição, onde um equipamento de sensoriamento é utilizado para obter os dados do mundo real (por exemplo, um tomógrafo). Tipicamente, os algoritmos de visualização volumétrica não fazem qualquer distinção quanto a origem ou natureza do

dado. Uma vez obtido o volume, o processo de visualização é o mesmo, independente da maneira como o dado foi obtido.

Conforme mencionado anteriormente, o ambiente volumétrico pode conter além de dados puramente volumétricos, objetos sólidos delimitados por superfícies. Conforme será visto mais adiante, o uso simultâneo de representações volumétricas e não volumétricas (superfícies poligonais) requer um esforço adicional para os algoritmos de visualização volumétrica, principalmente para resolver os problemas de *aliasing* (devido a erros durante o processo de amostragem dos *voxels* e polígonos) e visibilidade nas regiões de interseção entre *voxels* e polígonos.

O dado volumétrico pode ser facilmente compreendido como sendo uma função espacial. Entretanto, objetos sólidos também podem ser matematicamente modelados de maneira análoga. Nesse caso, pode-se definir uma função característica dos sólidos. Esta função é binária e indica a presença ou não do sólido em cada ponto do espaço. No entanto, a função característica tem algumas limitações, principalmente por não definir diretamente a estrutura do sólido (interior e fronteira) e não indicar o vetor normal à superfície do sólido. A fronteira do sólido pode ser indiretamente obtida, pois corresponde aos pontos onde a função característica é descontínua.

2.1 Modelo Conceitual

A Figura 2.1 ilustra o modelo conceitual de um dado volumétrico. No mundo físico temos dados espaciais, que constituem um sinal tridimensional representando a variação de uma propriedade física do dado. Essa propriedade, na maioria dos casos (em especial em imagens médicas), é uma grandeza puramente escalar, tipicamente representando a densidade, temperatura, pressão etc. nos diversos pontos do volume. Entretanto, a propriedade também pode ser uma grandeza multidimensional (tal como um vetor) representando, por exemplo, a velocidade em cada ponto do volume. A visualização de grandezas multidimensionais requer a utilização de técnicas especiais, sendo ainda uma área em estudo em visualização científica. Além disso, o ambiente volumétrico citado pode incluir também objetos sólidos, tipicamente delimitados por superfícies.

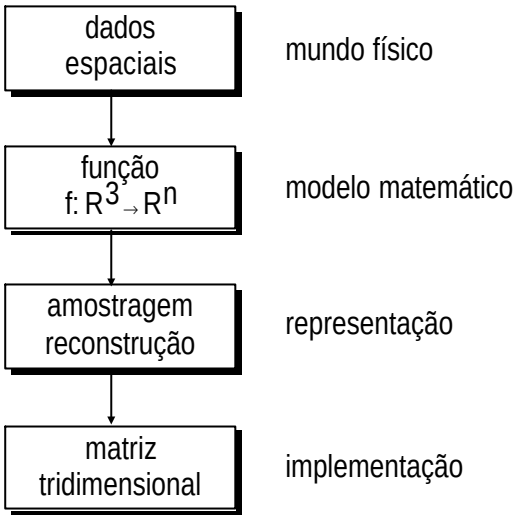


Figura 2.1: Modelo conceitual de um dado volumétrico

O modelo matemático capaz de expressar os dados volumétricos no mundo físico é baseado em funções do espaço. Este modelo é válido tanto para o dado volumétrico propriamente dito como também para os objetos sólidos existentes no ambiente. Um dos grandes desafios dessa área é a unificação das duas representações. Como será visto adiante, esta unificação é conceitualmente simples no modelo matemático.

Para representar tais funções é necessário realizar a amostragem do dado contínuo e a reconstrução de um dado contínuo a partir de um conjunto de dados discreto. Por razões de simplicidade, a amostragem tipicamente é uniforme, o que gera um conjunto de dados regular (reticulado). Esse passo é fundamental pois precisa-se obter uma representação discreta dos dados espacias. Entretanto, é importante ressaltar que pode haver perda de informação durante esse processo. Para que a perda seja controlada é necessário que a taxa de amostragem seja superior à freqüência de Nyquist. [Gomes e Velho, 97]

Por fim, no nível de implementação, este reticulado precisa ser armazenado em uma estrutura de dados apropriada. Caso a amostragem tenha sido uniforme, a estrutura mais adequada é uma matriz tridimensional. Esta forma é bastante compacta e permite rápido acesso a qualquer dado no espaço.

2.2 Distribuição Espacial dos Dados

Em geral, o dado volumétrico é amostrado ao longo de três eixos ortogonais, definindo um reticulado cartesiano. Esse reticulado pode ser regular, onde os intervalos são idênticos em todas as direções, ou retilíneo, onde os intervalos podem ser diferentes em cada direção, porém constantes em uma mesma direção. Entretanto, isso nem sempre acontece. Em algumas aplicações, os dados são amostrados em pontos aleatórios, o reticulado pode ser esférico, curvo etc. Essa generalização torna os algoritmos de visualização volumétrica extremamente ineficientes e, sempre que possível, deve ser evitado.

Uma classificação bastante utilizada para descrever a estrutura espacial do dado volumétrico pode ser encontrado em [Speray e outros, 90; Paiva e outros, 99], descrita resumidamente em sete categorias distintas:

- *Cartesiano*. Os elementos de dados são cubos alinhados com os eixos. É a forma ideal de estrutura espacial, pois permite acesso mais rápido aos dados;
- *Regular.* Os elementos são paralelepípedos de dimensões constantes. Esse tipo de estrutura é muito comum em dados provenientes de Tomografia Computadorizada;
- *Retilíneo*. Os elementos são paralelepípedos de dimensões variáveis;
- *Estruturado ou curvilíneo*. Os elementos não são alinhados com eixos e podem ser curvos. Muito comum em dinâmica de fluidos;
- *Estruturado em blocos*. Os dados são agrupados de forma a evitar limitações topológicas;
- *Não estruturado*. Sem quaisquer restrições geométricas. Muito comum em estudos de elementos finitos.

Um dos maiores problemas encontrados na visualização de volumes é a grande quantidade de dados. Uma matriz de 512x512x512 contendo dados escalares de apenas 1 byte cada necessita de 128 Mbytes para ser armazenada. Esse problema trouxe sérias limitações para a visualização interativa, pois o tempo necessário para o *rendering* do volume ainda era muito grande.

3 REPRESENTAÇÕES E CONVERSÕES

Em um ambiente onde existe apenas dados volumétricos, muitas vezes é interessante extrair superfícies relacionadas com o dado. No caso de imagens médicas, algumas estruturas particulares (por exemplo, pulmão, artéria etc) podem ser segmentadas, dando origem a superfícies e objetos sólidos. Em outras situações, como a apresentada na Figura 1.2, é necessário que primitivas geométricas sejam posteriormente adicionadas ao ambiente volumétrico. Em ambos os casos, é imprescindível a utilização de duas representações diferentes para modelar o ambiente. Por isso, é importante estudar com mais detalhes como essas representações estão relacionadas e como pode-se converter uma representação volumétrica em superfície e vice-versa.

3.1 Representação de Superfícies e Sólidos

A geometria de um objeto (superfície ou sólido) pode ser descrita de duas formas diferentes: utilizando o modelo paramétrico ou implícito [Gomes e Velho, 92]. No modelo paramétrico, o objeto é definido através de uma função que, variando seus parâmetros dentro do intervalo estabelecido, enumera todos os pontos do objeto. Por exemplo, um círculo unitário em 2D pode ser descrito na forma paramétrica através da função:

$$f(\theta) = (\cos\theta, \sin\theta),\ \theta \in [0, 2\pi]$$

Pode-se também definir a mesma geometria utilizando modelo implícito. Através desse esquema, o conjunto de pontos de define o objeto é especificado indiretamente, através de uma função de classificação. No exemplo do círculo, a forma implícita pode ser definida através da função:

$$F(x,y) = x^2 + y^2 - 1,\ x,y \in \Re$$

No modelo implícito, os pontos que pertencem ao círculo são todos aqueles em que $F(x,y)=0$. Nota-se que F é uma função que classifica os pontos do espaço com

relação ao objeto, definindo, portanto, uma subdivisão do espaço. A Figura 3.1 ilustra as diferenças entre as duas formas de descrição. Nesse caso, tem-se:

`F(x,y) < 0, (x,y)` pertence ao interior do círculo;

`F(x,y) = 0, (x,y)` pertence ao círculo;

`F(x,y) > 0, (x,y)` está do lado de fora do círculo.

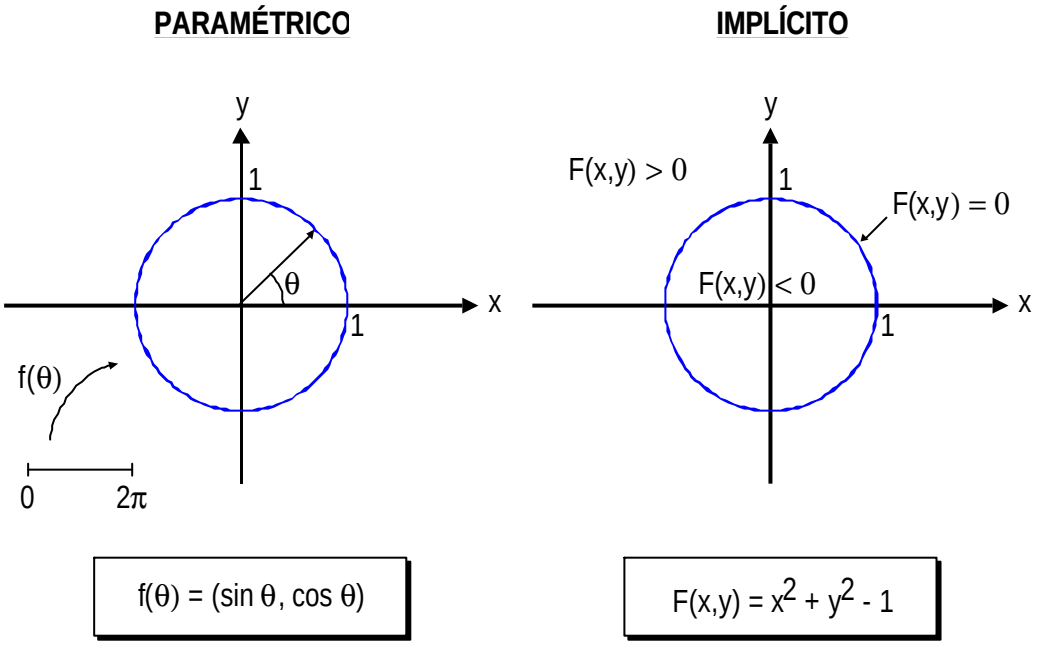


Figura 3.1: Descrição paramétrica e implícita

Pode-se dizer que o modelo paramétrico é muito mais usual, entretanto o modelo implícito está sendo cada vez mais estudado, tendo em vista sua importância para a modelagem geométrica [Gomes e Velho, 92]. O modelo paramétrico fornece uma representação explícita da superfície, porém tem algumas limitações. Uma delas é que, em algumas situações, é difícil encontrar um espaço de parâmetros adequado. Por isso, muitas vezes é necessário representar o espaço paramétrico por partes.

É importante ressaltar, no entanto, que as duas formas são complementares. Assim, uma pode ser mais adequada que a outra em determinadas situações. Para exemplificar, quando é preciso desenhar um objeto, o modelo paramétrico é o mais indicado, pois para obter todos os seus pontos basta variar os parâmetros. Entretanto, problemas típicos de

modelagem como, por exemplo, deteção de interferência entre objetos, é muito mais simples utilizando a formulação implícita.

Todos os conceitos apresentados aqui têm uma forte relação com visualização volumétrica. A partir do modelo volumétrico geral, definido através de uma função ou sinal tridimensional, pode-se obter um modelo volumétrico mais especializado como, por exemplo, baseado na formulação implícita dos objetos.

Esta formulação tem várias vantagens na resolução de problemas de modelagem geométrica. Por exemplo, o gradiente de uma função implícita pode ser utilizado para o cálculo do vetor normal a superfície. Além disso, a formulação implícita está intimamente relacionada com as isosuperfícies (curvas de nível) de uma função de densidade. Considerando o objeto representado por esta função implícita, sua fronteira também é uma isosuperfície e corresponde exatamente à região onde a densidade sofre uma variação mais acentuada. Isso é muito importante para as tarefas de visualização. Por último, pode-se também obter a vizinhança tubular de um sólido implícito, o que é extremamente útil para o cálculo de interferências e resolução de diversos problemas de modelagem geométrica.

Um objeto sólido ϑ também pode ser representado através de sua função característica, que vale 1 em todos os pontos pertencentes ao objeto e 0 em todos os demais pontos do espaço. Em outras palavras, seja $\chi_\vartheta: \Re^n \rightarrow \{0,1\}$ a função característica do objeto ϑ. Portanto,

$\chi_\vartheta(p) = 1$ se $p \in \vartheta$ e

$\chi_\vartheta(p) = 0$ se $p \notin \vartheta$.

A função característica de um sólido pode ser utilizada para criar uma descrição implícita do objeto. Note que a fronteira do objeto corresponde à região onde sua função característica é descontínua e também corresponde à uma curva de nível no modelo implícito.

Pode-se converter facilmente de uma representação implícita F para a função característica χ_ϑ através de:

$\chi_\vartheta(p) = 1$ se $F(p) \leq 0$ e

$\chi_\vartheta(p) = 0$ se $F(p) > 0$.

O caso inverso, i.é., conversão de uma função característica em uma representação implícita não é trivial e pode ser encontrado em [Velho e outros, 95]. Este procedimento é baseado em um método de deteção de borda e reconstrução utilizando *wavelets.* Este método é interessante pois também pode ser utilizado para obter uma conversão aproximada de uma representação paramétrica para implícita. Para isso, primeiro é obtido a função característica do objeto. Isso é realizado através da rasterização volumétrica do objeto (rasterizando primeiro a superfície e depois preenchendo o volume definido por ela) [Kaufman, 87; Pavlidis, 78]. Existem também métodos algébricos exatos para conversão de paramétrico em implícito, porém são muito complexos e dificilmente utilizados.

3.2 Métodos de Conversão

A criação de representações discretas a partir de objetos implícitos é um problema bastante importante em Visualização Científica. Uma das formas mais gerais de representação implícita é exatamente dados volumétricos, armazenados em uma matriz tridimensional contendo valores escalares, ou seja, uma função ou sinal 3D. A reconstrução de uma função contínua a partir de um conjunto discreto de pontos pode ser muito útil para resolver vários problemas de modelagem em Visualização Científica.

3.2.1 Poligonização

Os métodos de discretização geométrica procuram gerar, a partir de um modelo geométrico contínuo, uma representação discreta aproximada do modelo original. De uma forma geral, os métodos de poligonização podem ser comparados com o problema de lapidação de uma pedra preciosa, onde é necessário esculpir faces planas na pedra com mínima perda de material [Gomes e Velho, 92].

Tipicamente, o primeiro passo de qualquer algoritmo de poligonização é discretizar o domínio da função, gerando assim, uma decomposição celular do espaço ambiente. Em muitas situações (por exemplo, dados volumétricos) o domínio já está discretizado. Nesse caso, quando a função original não é conhecida, a aproximação poligonal pode não ser bem definida.

O segundo passo é gerar uma representação paramétrica linear por partes, ou seja, uma malha de polígonos. Para isso, é necessário classificar as células geradas a partir da

discretização do domínio. A Figura 3.2 ilustra o caso bidimensional, porém o mesmo pode ser aplicado em 3D. Através da análise do sinal da função implícita, pode-se inferir se a célula está dentro (Figura 3.2a), fora (Figura 3.2b) do sólido implícito ou tem interseção com a superfície implícita (Figura 3.2c). Se o sinal da função implícita não é o mesmo em todos os vértices da célula, então há interseção da célula com o sólido implícito. Entretanto, deve-se ter cuidado para não utilizar uma discretização inadequada do espaço ambiente, como ilustra a Figura 3.2d.

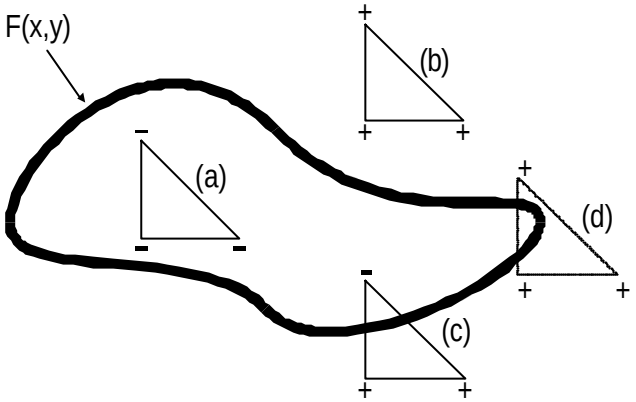


Figura 3.2: Classificação das células

Após a classificação de todas as células, é necessário gerar os polígonos, calculando-se a interseção da fronteira de cada célula com a superfície implícita. Para agilizar o processo, normalmente, utiliza-se uma tabela com todas os casos possíveis de interseção, já que o número de possibilidades não é muito grande. A seguir, os polígonos são “colados”, ou seja, reconstruídos com estruturação. Com isso, obtem-se uma descrição paramétrica por partes da função implícita.

Com relação à decomposição do domínio, os algoritmos de poligonização podem ser classificados como simpliciais e não simpliciais. Os simpliciais triangulam o domínio da função implícita. No caso 3D, as células geradas são tetraedros. Os algoritmos não simpliciais tesselam o domínio, gerando células mais complexas.

O exemplo mais conhecido de algoritmo não simplicial é *Marching Cubes* [Lorensen e Cline, 87]. Este método utiliza uma decomposição do volume em paralelepípedos, formados por cada conjunto de oito vértices adjacentes dentro do volume. Considerando um determinado valor de limiar o algoritmo procura identificar de

que forma a iso-superfície passa pelos cubos. A princípio, 256 (2^8) casos poderiam ser identificados. Entretanto, através simetrias, esse número pode ser reduzido para 16 casos básicos. Na verdade, são 15, pois o primeiro e último geram a mesma geometria (Veja Figura 3.3).

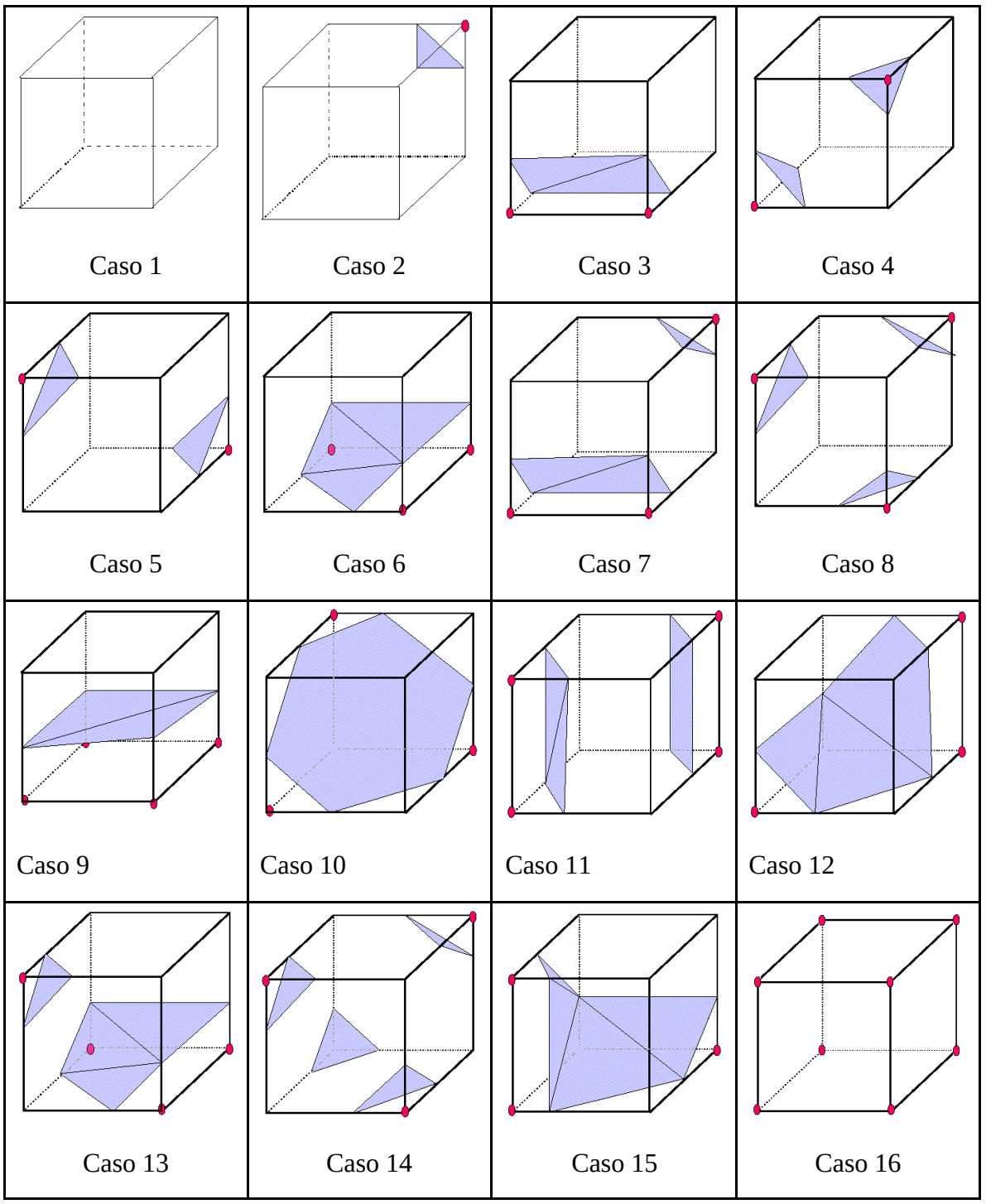


Figura 3.3: Casos básicos do algoritmo *Marching Cubes*

Os métodos simpliciais [Gomes e Velho, 92] possuem muito menos casos, pois a interseção, se houver, pode ocorrer em apenas duas situações diferentes, conforme mostrado na Figura 3.4. Além disso, os métodos simpliciais têm outras vantagens em relação aos não simpliciais. A mais importante é, sem dúvida, a ausência de ambigüidades. Muitas vezes, os algoritmos não simpliciais como o *Marching Cubes*, têm que resolver ambigüidades com coerência, tomando decisões *ad-hoc*, para evitar que aparecam furos na representação poligonal. Os métodos simpliciais, por outro lado, geram um número maior de células após a decomposição do domínio, entretanto, os casos são mais simples de serem tratados e, no final da poligonização, o número de faces geradas é praticamento o mesmo.

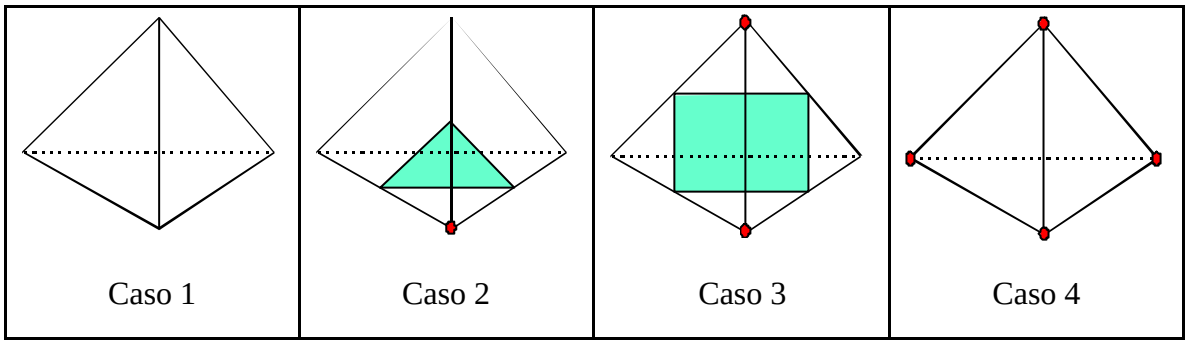


Figura 3.4: Casos básicos do método simplicial

Em todos os métodos de poligonização, é necessário uma classificação do dado volumétrico em duas regiões para identificar as iso-superfícies. Isso significa que, invariavelmente, muita informação é perdida durante esse processo de classificação. A principal conseqüência desse fato é que pequenos detalhes do volume podem ser ignorados. Além disso, muitos objetos amorfos (de natureza essencialmente volumétrica, tais como nuvem e fluidos), não podem ser representados satisfatoriamente utilizando apenas primitivas geométricas. Nesse caso, outras representações mais adequadas devem ser escolhidas.

3.2.2 Voxelização

Essa técnica permite converter um sólido dado por uma representação de fronteira (por exemplo, B-Rep) em uma representação volumétrica implícita. Com isto, um objeto

geométrico com uma representação geométrica contínua pode ser transformado em um conjunto discreto de *voxels* que mais se aproxima da representação contínua original.

Como o resultado desta conversão é essencialmente um conjunto volumétrico de dados representando diferentes densidades (assim como os resultados obtidos de simulações numéricas e exames médicos), ele pode ser visualizado através de qualquer algoritmo de visualização volumétrica. Com isso, pode-se facilmente misturar dados de natureza essencialmente volumétrica (por exemplo, um exame de ressonância magnética) com modelos geométricos, já que, após a conversão do último, tem-se uma representação comum. Obviamente, em muitas situações, essa conversão em uma representação única não é desejada, visto que sempre há perdas.

O processo de conversão de um modelo de fronteira em volumétrico tem uma forte analogia com a conversão de um objeto geométrico 2D em *pixels*. Entretanto, o processo de conversão em 3D, também chamado de *3D scan conversion*, é mais complexo. Em [Kaufman, 87] pode ser encontrado um estudo de alguns desses algoritmos. A princípio, pode-se achar que basta selecionar todos os *voxels* que pertencem total ou parcialmente ao objeto. Porém, o resultado obtido normalmente não é satisfatório, pois a conversão é muito grosseira. Isso ocorre porque a qualidade de uma classificação binária dos *voxels* está diretamente relacionada com a resolução espacial do volume. Por isso, o *aliasing* é inevitável quando não se toma os devidos cuidados.

Em [Wang e Kaufman, 93] pode ser encontrado uma técnica para reduzir o efeito de *aliasing*, baseado na estimativa da contribuição do sólido na densidade dos *voxels*. Com isso, a densidade de um *voxel* é proporcional a distância de seu centro ao objeto sólido. Isso pode ser visto como um filtro que suaviza a função característica do sólido, reduzindo, assim, o *aliasing*. [Breen e outros, 88] também descrevem, entre outras assuntos, um método de *scan conversion* de modelos CSG. Essa técnica é utilizada para gerar um volume dados, representado por uma função implícita, onde cada *voxel* armazena a menor distância entre a superficie do sólido e o *voxel* (função de distância).

4 VISUALIZAÇÃO VOLUMÉTRICA

A visualização de dados volumétricos é um caso mais geral que a visualização de superfícies. A característica básica de um dado volumétrico é, muitas vezes, não apresentar superfícies explícitas, ou seja, as fronteiras não são muito bem definidas, os objetos podem ser amorfos. Isto pode ocorrer em diversos graus, desde objetos pouco amorfos, tais como sólidos comuns com fronteira bem definida, até objetos totalmente amorfos, tais como nuvens, gases etc. No último caso, não é possível identificar as superfícies que formam o objeto.

Uma outra questão importante está relacionada com o meio onde os objetos estão inseridos. Este meio pode ser não participativo ou participativo. O meio não participativo é caracterizado por não atenuar a luz durante sua propagação. Esse é o caso mais simples, pois a iluminação precisa ser calculada apenas nas fronteiras dos objetos. O caso mais geral, quando o meio é participativo, é muito mais complexo e a visualização exata da cena, segundo os modelos físicos, é computacionalmente difícil.

No caso da visualização volumétrica, não se espera a perfeita simulação da realidade, mas sim a visualização exploratória dos dados. Além disso, o tempo para a geração de uma imagem deve ser razoável o suficiente a ponto de permitir o mínimo de interatividade. Por esse motivo, simplificações devem ser realizadas no modelo físico da propagação da luz. Pode-se dizer que a visualização volumétrica é uma situação intermediária, entre meio não participativo e meio participativo.

De uma maneira bem simples, a visualização de um dado volumétrico típico pode ser vista como sendo a visualização de sólidos dentro de sólidos, cuja função densidade é constante dentro do volume. Entretanto, esses objetos sólidos podem ter, na verdade, fronteiras difusas. O que caracteriza uma fronteira difusa é a variação suave da função densidade numa vizinhança tubular da fronteira. Isso pode ser estimado através da magnitude do gradiente da função. Quando a magnitude é grande, significa que há uma superfície bem nítida separando o meio. Quando é pequena, significa uma superfície difusa sendo, portanto, muito difícil caracterizar o objeto.

Uma das simplificações mais comuns realizadas no modelo físico é admitir que há pouco espalhamento da luz quando ela se propaga pelo volume. Nesse caso, o meio pode

ser modelado como um gelatina semi-transparente. A luz, ao percorrer o meio, interage com a gelatina, podendo ser absorvida, emitida ou espalhada (Veja a Figura 4.1). Nesse caso, como há interação com o meio, o cálculo da iluminação tem que realizado continuamente sendo, portanto, necessário integrar o efeito da atenuação da luz durante a sua propagação pelo meio em que se encontra. Uma outra simplificação muito comum é não considerar a atenuação da luz emitida pelas fontes luminosas pelo meio.

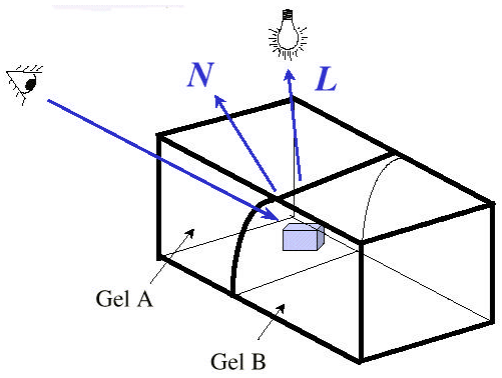


Figura 4.1: Propagação da luz no ambiente volumétrico [Paiva e outros, 99].

Todas essas simplificações reduzem bastante a complexidade dos algoritmos de visualização, diminuindo o tempo de geração das imagens e permitindo a visualização exploratória dos dados. Isso, obviamente, causa um pouco de perda na qualidade das imagens. Porém, para a maioria das aplicações (principalmente em medicina, onde não há tanta necessidade de visualização de imagens reais), o ganho de velocidade justifica a pequena perda de qualidade.

4.1 *Pipeline* de Visualização

Os algoritmos de visualização volumétrica requerem, em sua maioria, os passos apresentados na Figura 4.2. O primeiro passo consiste na aquisição do dado volumétrico, que normalmente é realizado através do uso de um equipamento de sensoriamento apropriado. A seguir, o dado volumétrico bruto é classificado, normalmente com o auxílio do usuário. Essa etapa pode ser vista como uma forma de segmentação, pois sua finalidade principal é identificar estruturas internas do volume, através do uso adequado de cores e opacidades. A seguir, o volume classificado é iluminado, facilitando sua

interpretação pelo usuário, pois a forma tridimensional do volume é realçada. O último passo consiste na projeção do volume classificado e iluminado em um plano de visualização, gerando assim uma imagem bidimensional que será visualizada pelo usuário.

Essas etapas são muito comuns na grande maioria dos algoritmos de visualização. Nessa análise, está sendo considerado que o dado volumétrico é apresentado sob a forma de valores escalares em um reticulado retilínio. Esta etapa já foi previamente realizada através de alguma simulação numérica ou através de algum processo de aquisição.

Tipicamente, esses dados podem ser pré-processados, visando reduzir o ruído, realçar algumas características do objeto em estudo. Em algumas situações, os dados precisam ser reconstruídos, visando estimar valores perdidos durante o processo de aquisição, reamostrar o volume, através de interpolação, para adequar sua topologia (convertendo um reticulado irregular para um reticulado regular, por exemplo), novas fatias podem ser criadas e outras podem ser descartadas etc.

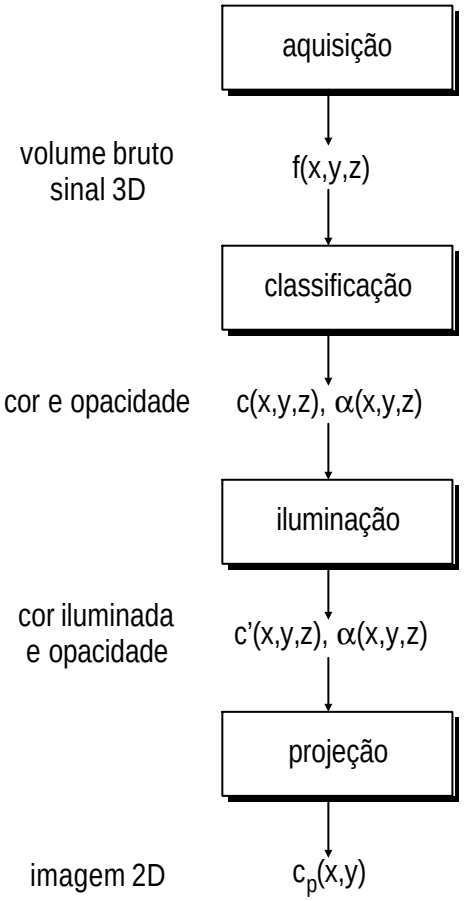


Figura 4.2: Pipeline de visualização volumétrica.

Pressupondo que o dado volumétrico já esteja armazenado em uma estrutura adequada, o processo de visualização volumétrica consiste em, a partir de uma função escalar discreta $f(x_i, y_j, z_k)$ que define o volume, gerar uma imagem bidimensional $p(x_i, y_j)$ que represente a projeção desse volume em um determinado plano, de acordo com os parâmetros de visualização.

O interessante ao estudar o *pipeline* de visualização é perceber que em cada etapa uma novidade é introduzida. Após a aquisição dos dados, tem-se o volume em seu estado bruto. Ao separar as estruturas que serão visualizadas, através da definição das funções de transferências, tem-se o dado classificado. Após o posicionamento das luzes, tem-se o dado iluminado. Finalmente, após o posicionamento da câmera, tem-se o dado projetado em uma imagem bidimensional. Fica claro, então, que qualquer alteração nesse processo é necessário revisitar as etapas seguintes. Por exemplo, se a posição da câmera for alterada, apenas a etapa de projeção precisa ser executada. Entretanto, se o usuário mudar uma das funções de transferência, as etapas de classificação, iluminação e projeção precisam ser novamente executadas.

4.1.1 Classificação

A classificação do volume é etapa mais complexa para o usuário de um sistema de visualização volumétrica (por exemplo, um médico). Normalmente, esse usuário não tem conhecimentos técnicos a respeito do processo de visualização, entretanto o requisito fundamental para uma boa classificação do volume é o conhecimento prévio do objeto em estudo, i.é., o usuário deverá reconhecer sua anatomia, as estruturas que poderão eventualmente serem encontradas, em quais posições etc. [Elvins, 92].

O processo de classificação tem a finalidade principal de identificar as estruturas internas do volume. Com isso, um médico pode, por exemplo, isolar elementos distintos, tais como gordura, músculo e osso, em um exame de ressonância magnética. No caso de algoritmos de *rendering* direto, cujos dados são representados por grandezas escalares, a classificação do volume envolve tipicamente a definição de funções de transferência de cor e opacidade.

A função de transferência de cor permite mapear valores contidos no volume em uma cor RGB característica. Por exemplo, em um exame de tomografia computadorizada, pode-se definir uma faixa de intensidade como sendo gordura e

associar a cor amarela. Para uma outra faixa, pode-se definir como sendo músculo e atribuir a cor vermelha. Osso pode ser representado pela cor branca etc (Veja a Figura 4.3).

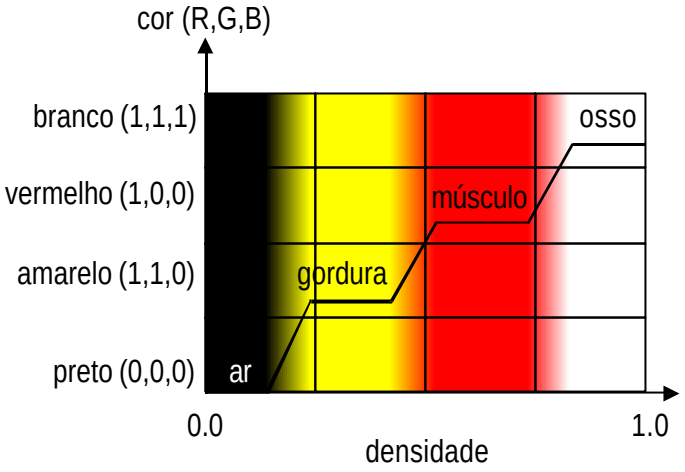


Figura 4.3: Função de transferência de cor

A função de transferência de opacidade permite mapear valores contidos em um volume em uma determinada transparência, normalmente representada por uma número real no intervalo de 0.0, para totalmente transparente, a 1.0, para totalmente opaco (Veja a Figura 4.4).

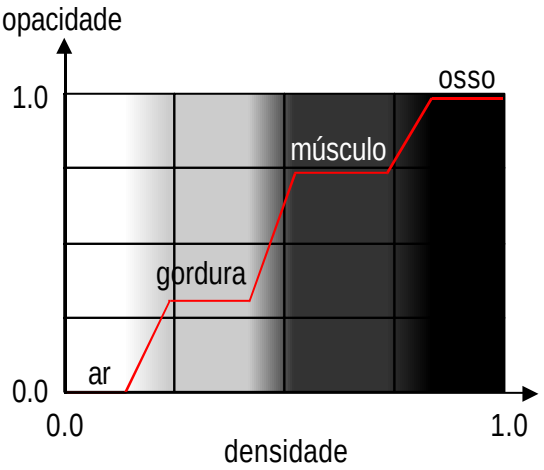
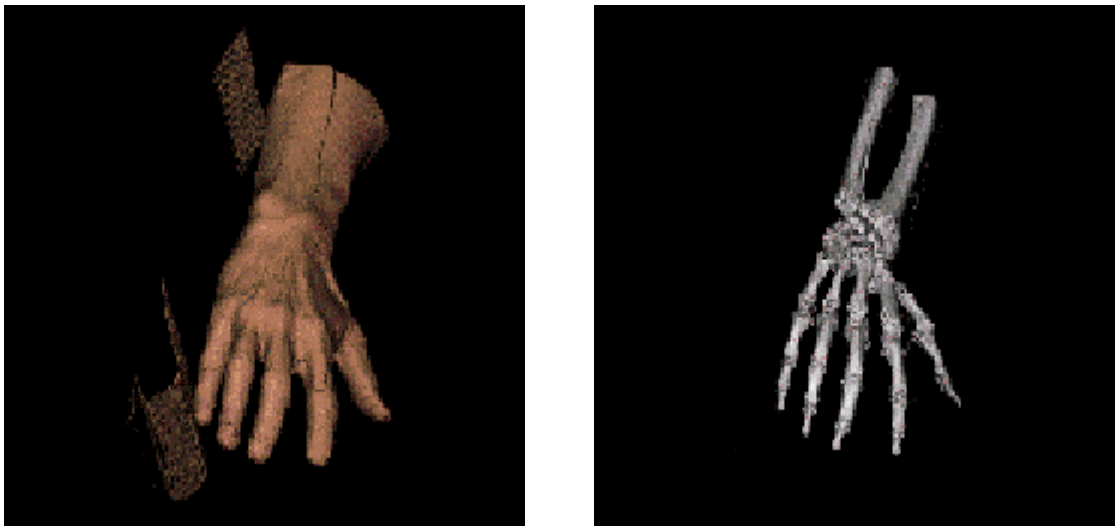


Figura 4.4: Função de transferência de opacidade

A modificação da função de transferência de opacidade permite ao usuário visualizar determinadas estruturas do volume. Para isso, basta torná-las não transparentes e deixar as demais transparentes, ou quase transparentes. Tipicamente, quando se trata de imagens médicas, as camadas mais externas do volume são colocadas mais transparentes que as mais internas. Com isso, o usuário (por exemplo, um médico) pode ter uma visualização das partes mais internas do volume. (Veja a Figura 4.5).



(a) (b)

Figura 4.5: Mudança nas funções de transferência [Paiva e outros, 99]

Existem algumas técnicas que permitem a classificação automática do volume. Uma delas é muito utilizada em exames de tomografia computadorizada e baseia-se em um modelo de classificação probabilístico [Drebin e outros, 88]. Nesse modelo, é assumido que apenas um ou no máximo dois elementos podem existir em um *voxel*. Além disso, a existência de dois elementos só é possível na interface que os separam, por exemplo, a região entre músculo e osso.

Apesar de existirem outras técnicas automáticas de classificação de volumes, na maioria dos casos ainda é necessária a interferência do usuário para definir adequadamente as funções de transferências. O método utilizado para este fim ainda é o da tentativa e erro. Obviamente, existem ferramentas capazes de auxiliar o usuário nesse processo. Uma das mais importantes é o histograma do volume, onde é apresentado a distribuição dos valores escalares que ocorrem no volume. Com isso, o usuário pode ter uma idéia mais precisa das estruturas ali contidas.

4.1.2 Iluminação

Para criar uma ilusão de profundidade, realçar bordas e características do volume, é muito comum a utilização de um modelo de iluminação para a visualização do volume. Os algoritmos de visualização volumétrica tipicamente utilizam aproximações de modelos de iluminação de baixo espalhamento, pois são computacionalmente mais simples e apresentam bons resultados, principalmente com imagens médicas. A inclusão de modelos de alto espalhamento pode proporcionar maior realismo para a cena, porém, no caso de dados médicos, não traz grandes benefícios para sua interpretação. Portanto, na maioria dos casos, não justifica a utilização de modelos de alto espalhamento.

A aproximação de um modelo de iluminação de volumes de baixo espalhamento é realizado através de um modelo local de iluminação. Os modelos locais consideram apenas a reflexão da luz na superfície do objeto. Isso significa que, nesses modelos, apenas a luz ambiente e as reflexões difusa e especular das fontes de luz existentes na cena são consideradas no cálculo da iluminação.

Entre os modelos locais de iluminação, os mais conhecidos são Gouraud [Gourad, 71] e Phong [Phong, 75]. No método de Gourad, a iluminação é calculada nos vértices da célula e interpolada em seu interior. Já no método de Phong, a normal nos pontos dentro de uma célula é calculada através da interpolação das normais nos vértices da mesma. A iluminação é, portanto, calculada em cada ponto no interior da célula.

No caso de algoritmos de *rendering* direto, o conceito de superfície pode não fazer muito sentido, mas é de extrema importância para o correto cálculo da iluminação dos *voxels*. Nessa situação, imagina-se que o ponto sendo iluminado pertence a uma isosuperfície existente no volume. Efetivamente, os algoritmos procuram fazer uma estimativa da normal à superfície para efeito de iluminação.

Um dos métodos mais utilizados para a estimativa da normal em um determinado ponto do volume é utilizando uma aproximação do gradiente do volume. Essa aproximação é calculada através de diferenças finitas em uma vizinhança do campo escalar volumétrico. Utilizando diferenças centrais, a normal **`N`** pode ser estimada pela fórmula abaixo: (`D[i,j,k]` representa o valor escalar na posição `(i,j,k)` do volume)

N_x `= D[i+1,j,k] - D[i-1,j,k]`

N_y `= D[i,j+1,k] - D[i,j-1,k]`

N_z `= D[i,j,k+1] - D[i,j,k-1]`

Essa estimativa da normal é bastante simples de ser realizada, porém, por ser baseada em uma vizinhança muito próxima do volume, é muito sensível a ruídos e descontinuidades do volume. Em algumas situações, a estimativa não pode ser utilizada. Para reduzir o efeito da localidade, pode-se estimar o gradiente considerando não apenas os seis vértices vizinhos, mas sim os 26 vizinhos (estimativa de segunda ordem). Apesar de ser um método bastante simples, a estimativa da normal pode consumir muito tempo de computação, pois deve ser feito em todos os pontos do volume. Por isso, esse cálculo é normalmente realizado como uma etapa de pré-processamento do volume.

4.1.3 Projeção

Essa etapa no *pipeline* de visualização recebe o volume já iluminado e realiza a projeção dos *voxels* sobre o plano da imagem. Tipicamente, a projeção é realizada através do lançamento de um raio a partir de cada *pixel* da tela (plano de projeção), determinando, através da composição da cor iluminada e opacidade dos *voxels* atingidos 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 mais graves é o problema da divergência dos raios. A densidade de raios, i.é., o número de raios lançados por *voxel*, diminui a medida que se caminha sobre o raio. Ou seja, a amostragem realizada nos *voxels* mais próximos do observador é mais detalhada que a realizada nos *voxels* mais distantes. Isso significa que pequenos detalhes são perdidos ao serem projetos os *voxels* mais distantes. Para resolver este problema, pode-se usar a força bruta, aumentando o número de raios lançados (*oversampling*). Obviamente, este método é bastante ineficiente. Uma outra forma é utilizar um algoritmo adaptativo, aumentando-se a densidade de raios a medida que se afasta do observador [Novins e outros, 90].

4.2 Algoritmos de Visualização

Essa seção tem o objetivo de apresentar os algoritmos básicos de *rendering* direto. Esses algoritmos têm em comum a não utilização de nenhuma representação intermediária para gerar a imagem do volume. Porém, têm a desvantagem de serem, em geral, computacionalmente mais caros.

Como exemplo de algoritmos de *rendering* direto, pode-se citar *Ray Casting* [Levoy, 88], *Splatting* [Westover, 90] e *Shear-Warp* [Lacroute, 95], entre outros. Esses algoritmos realizam a projeção do volume diretamente no plano da imagem e são os mais indicados para visualizar volumes que representam objetos amorfos.

Pode-se ainda subdividir os algoritmos de *rendering* direto em duas categorias principais: espaço dos objetos e espaço da imagem. Na primeira situação (espaço dos objetos), é utilizado o mapeamento direto (*forward mapping*) do volume sobre o plano da imagem, ou seja, os *voxels* são projetados na tela em uma determinada ordem. Como exemplo, pode-se citar os algoritmos *Splatting* [Westover, 90] e *V-Buffer* [Upson e Keeler, 88]. No outro caso (espaço da imagem), é utilizado o mapeamento reverso (*backward mapping*), onde são lançados raios para cada pixel da imagem em direção ao volume determinando, assim, a cor final do *pixel*. Como exemplo, pode-se citar o algoritmo *Ray Casting* [Levoy, 88].

A seguir serão apresentadas as principais características dos algoritmos mais importantes de *rendering* direto: o traçado de raios (*Ray Casting*), que opera no espaço da imagem, o *Splatting*, que opera no espaço do objeto e o algoritmo *Shear-Warp* que mescla as duas técnicas.

4.2.1 Ray Casting

O algoritmo *Ray Casting* [Levoy, 88] permite a geração de imagens de alta qualidade através de *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 4.6.

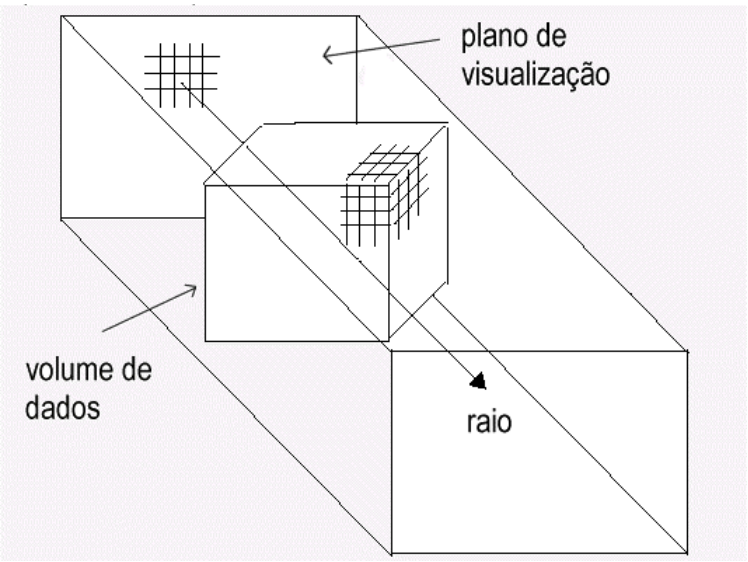


Figura 4.6: Algoritmo *Ray Casting*

Durante a etapa de iluminação do *pipeline* de visualização volumétrica, 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, tipicamente utilizando diferenças finitas. Além disto, algumas luzes podem ser posicionadas na cena. A Figura 4.7 ilustra esse processo.

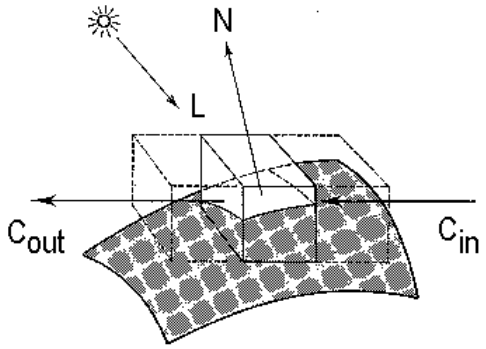


Figura 4.7: Cálculo da iluminação em um *voxel*

Pela figura acima, a luz C_{out} que sai de um *voxel* é calculada em função de três fatores fundamentais:

- luz refletida na direção de visão por todas as fontes de luz presentes na cena;

- luz C_{in} filtrada pelo *voxel*;
- qualquer luz emitida pelo *voxel*.

Tipicamente, a luz emitida pelas fontes de luz presentes na cena não é atenuada pelo volume. Esse efeito é usualmente ignorado durante o cálculo da iluminação pois, além de simplificar bastante o algoritmo (não é necessário calcular uma integral de linha), 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 obscurecidas. Note, entretanto, 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.

Para cada raio lançado no volume (Figura 4.6), é realizada uma amostragem do volume em intervalos regulares ao longo do raio. Através de interpolação, é obtida a cor e a opacidade em cada ponto amostrado, segundo as funções de transferências empregadas durante o processo de classificação. A seguir, todas as contribuições obtidas (cores e opacidades) são compostas calculando assim a cor final $C(R)$ do *pixel* (Figura 4.8).

A cor $C_{out}(R,k)$ e a opacidade $\alpha_{out}(R,k)$ do raio após passar pela amostra k ao longo do raio é calculada em função da cor $C_{in}(R,k)$ e a opacidade $\alpha_{in}(R,k)$ antes de passar pelo *voxel* e pela cor $C(R,k)$ e opacidade $\alpha(R,k)$ da amostra:

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

$$\alpha_{out}(R,k) = \alpha_{in}(R,k) + \alpha(R,k)*(1-\alpha_{in}(R,k))$$

Sendo que,

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

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

Depois de computar todas as N contribuições de $k=1..N$, a cor final $C(R)$ do *pixel* é dada por:

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

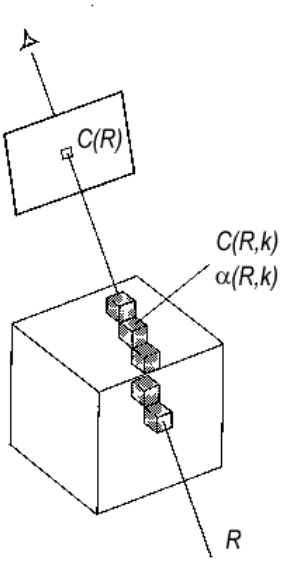


Figura 4.8: Composição de cor e opacidade

Essencialmente, esse cálculo pode ser expresso através de:

$$
\begin{aligned}
C(R) = {} & C(R,1) + \\
& C(R,2)*(1-\alpha(R,1)) + \\
& C(R,3)*(1-\alpha(R,1))*(1-\alpha(R,2)) + \\
& \ldots + \\
& C(R,N)*(1-\alpha(R,1))\ldots(1-\alpha(R,N-1)) \\
= {} & \sum_{k=1}^{N} \left(C(R,k) * \prod_{i=1}^{k-1} (1-\alpha(R,i)) \right)
\end{aligned}
$$

Pode-se também utilizar o operador *over* de composição digital [Porter e Duff, 84] para expressar a cor final do *pixel*:

$$C(R) = C(R,1) \ over \ C(R,2) \ over \ C(R,3) \ \ldots \ over \ C(R,N)$$

O algoritmo *Ray Casting* gera imagens de alta qualidade, entretanto seu custo computacional pode ser muito grande, dependendo da resolução final da imagem, i.é., da quantidade de raios a serem lançados pelo volume. Uma outra desvantagem é que os raios lançados tipicamente atravessam o volume fora da ordem de armazenamento dos

voxels, sendo necessário, portanto, calcular em que *voxel* está cada ponto amostrado ao longo do raio.

Existem várias formas de aceleração do algoritmo. Quando o processo de composição do raio é realizado de frente para trás, i.é., do ponto mais próximo do observador ao mais distante, pode-se interromper o processo quando a opacidade acumulada atingir um certo valor limite, fazendo com que os *voxels* mais afastadas não contribuam muito para a cor final do *pixel*. Além disso, pode-se utilizar *octrees* para rapidamente saltar por regiões homogêneas do volume (densidade constante), simplificando bastante o processo de integração ao longo do raio. Essas otimizações podem ser encontradas em [Levoy, 90b; Levoy 90c].

Além disso, pode-se explorar uma característica bem interessante do algoritmo: como os raios lançamento são totalmente independentes entre si, pode-se visualizar progressivamente a construção da imagem (Figura 4.9).

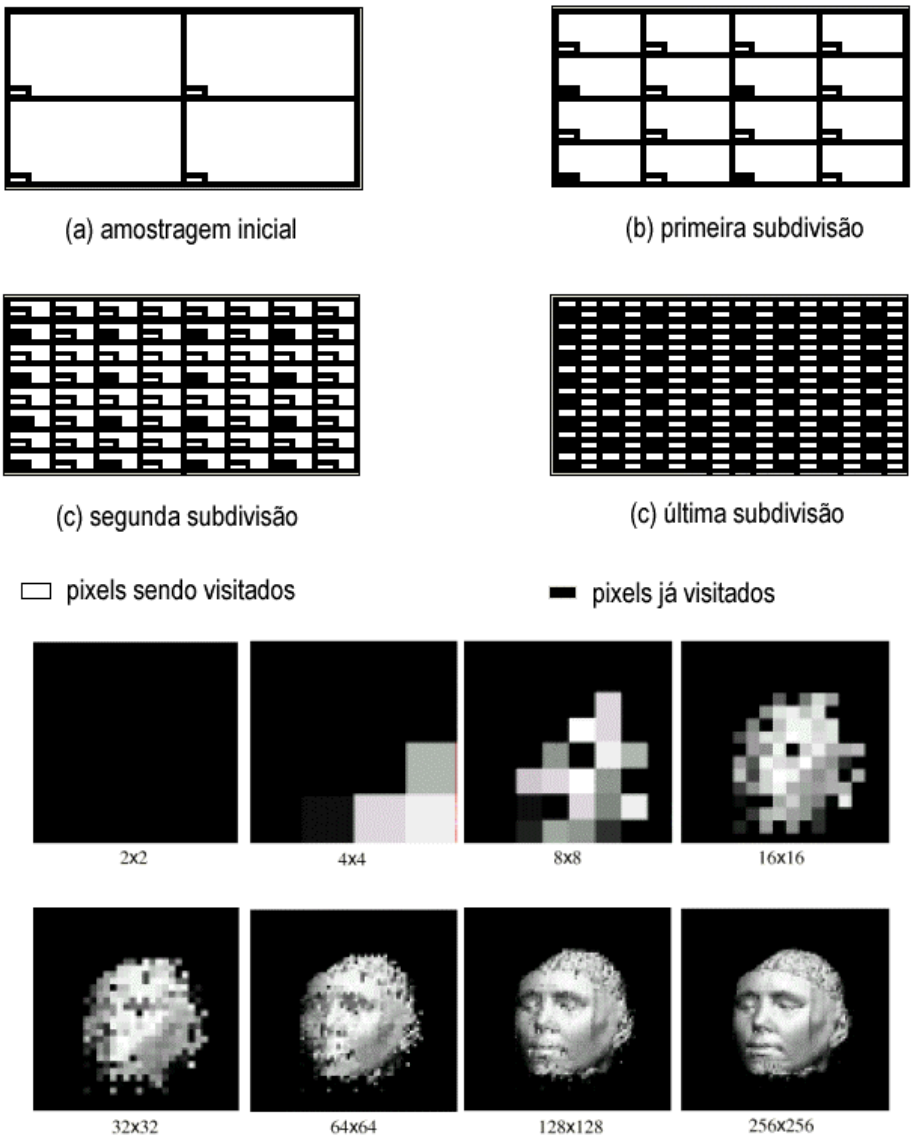


Figura 4.9: Refinamento progressivo [Paiva e outros, 99]

4.2.2 Splatting

O algoritmo *Splatting* [Westover, 90] 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 de 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, assim como no algoritmo *Ray Casting*, a ordem correta de projeção permite acumular adequadamente as

opacidades dos *voxels*. Para isso, escolhe-se os dois eixos do volume mais paralelos ao plano da imagem para formar o *loop* mais interno. Com isso, o plano formado por esses dois eixos serão rapidamente projetados e o usuário poderá perceber mais rapidamente a formação da imagem final. A projeção é realizada fatia por fatia, ou seja, todos os *voxels* de uma determinada fatia são projetados antes que a próxima fatia 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, conforme apresentado na seção 4.1.2.

A seguir vem a parte mais importante do algoritmo, onde 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 extensão da projeção do *voxel* sobre o plano da imagem. A projeção do *kernel* é chamada de *footprint* e, no caso de projeções ortográficas, 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*.

A 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 de frente para atrás, uma vantagem do algoritmo é permitir que o usuário acompanhe o processo de geração da imagem final. Isso é mais eficiente que, por exemplo, no algoritmo *Ray Casting*, pois é projetada uma fatia do volume de cada vez, e não um *pixel* por vez. Além disso, o algoritmo *Splatting* é facilmente paralelizável, pois a

projeção de cada *voxel* é realizada de modo independente, ou seja, não é necessário considerar o restante do volume durante a projeção. Entretanto, a ordem correta de projeção deve ser respeitada.

4.2.3 Shear-Warp

A idéia principal do algoritmo *Shear-Warp* [Lacroute, 95] é transformar o volume de dados de modo a simplificar a etapa de projeção do *pipeline* de visualização. Isto é obtido através da aplicação de um cisalhamento nas fatias do volume, transformando os dados para um sistema de coordenadas intermediário cuja principal caracterísica é que os raios de visão são perpendiculares as fatias do volume (Figura 4.10). 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 afim que simplesmente translada as fatias do volume, não sendo, portanto, computacionalmente cara.

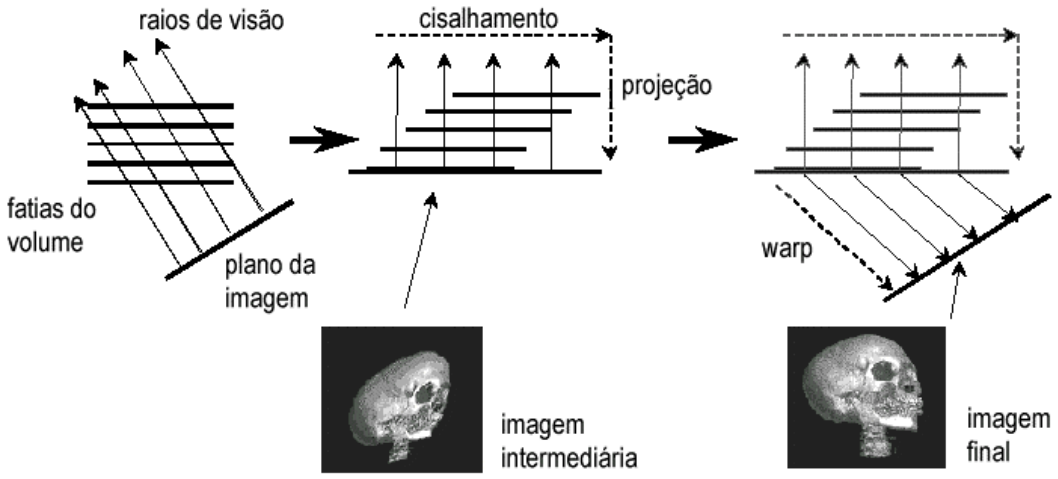


Figura 4.10: Algoritmo *Shear-Warp*

A composição e projeção das fatias nesse novo sistema de coordenadas (realizada através do operador *over*) gera 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 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.

Como a construção da imagem intermediária é realizada *scanline* a *scanline*, pode-se tirar proveito disso para otimizar ainda mais o algoritmo. A implementação original realizada por Lacroute utiliza estruturas do tipo RLE (*run length encoding*) para agilizar o processamento dos *voxels* nas fatias e *pixels* na imagem intermediária. Através do uso dessa codificação, gerada em uma etapa de pré-processamento, pode-se facilmente ignorar os *voxels* transparentes e os *pixels* já opacos (muito semelhante ao término precoce de raios utilizado no algoritmo *Ray Casting*). 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*.

5 VISUALIZAÇÃO VOLUMÉTRICA HÍBRIDA

Conforme mencionado nos capítulos anteriores, diversas aplicações científicas necessitam de visualização simultânea de representações volumétricas e geométricas. Algumas estratégias foram propostas para combinar essas duas representações e, de uma maneira bastante simplificada, elas podem ser classificadas de acordo com o tipo de problema a ser resolvido. Algumas estratégias estão preocupadas em resolver o problema da modelagem, enquanto outras procuram resolver o problema da visualização [Tost e outros, 93].

A primeira estratégia (modelagem) procura converter uma representação em outra, conforme apresentado no Capítulo 3. Entre as técnicas mais comuns, pode-se citar a voxelização e poligonização. Esse tipo de solução tem a vantagem de tratar representações diferentes da mesma forma, entretanto a grande desvantagem é a inevitável perda de informação após a conversão. No caso da poligonização, pequenos detalhes são perdidos e objetos amorfos não podem ser representados satisfatoriamente através de superfícies. Já os algoritmos de voxelização tem a desvantagem de discretizar a geometria, podendo assim, ser introduzido *aliasing*.

A segunda estratégia (visualização) procura gerar a imagem final diretamente da da cena contendo representações diferentes, sem utilizar nenhum tipo de conversão. Essa solução tem a vantagem de preservar as representações originais, ou seja, superfícies são tratadas como superfícies. Além disso, é a única capaz de tratar cenas complexas. O principal objetivo deste capítulo é descrever algumas das técnicas mais importantes utilizadas nos algoritmos verdadeiramente híbridos de visualização volumétrica.

5.1 Métodos de Conversão

Os métodos de poligonização são caracterizados por procurar aproximar iso-superfícies dentro do volume utilizando primitivas geométricas, tipicamente superfícies poligonais. Por isso, são chamados de algoritmos de extração de superfícies. Um dos algoritmos mais conhecidos (*Marching Cubes*) foi apresentado no Capítulo 4. Devido a facilidade de realizar *rendering* de polígonos (esta capacidade é comumente implementada com eficiência em *hardware*), os primeiros algoritmos de visualização volumétrica

desenvolvidos foram desse tipo. Entretanto, não é adequado considerar os algoritmos de extração de superfícies como sendo algoritmos de visualização volumétrica pois, essencialmente, o que está sendo visualizado são superfícies e não o volume propriamente dito.

Uma outra forma de visualização híbrida converte todos os objetos geométricos contidos na cena em *voxels*. A Figura 5.1 ilustra as etapas principais desse método. A rasterização dos objetos gera um conjunto de *voxels* espacialmente superpostos ao dado volume. A seguir, todos os *voxels* são combinados gerando assim um único conjunto de dados volumétricos.

Para calcular a cor C_C e opacidade α_C resultante da combinação de dois *voxels* cujas cores e opacidades são (C_G, α_G) e (C_V, α_V) respectivamente, pode-se utilizar a fórmula:

$$C_C = (C_G\alpha_G + C_V\alpha_V) / (\alpha_G + \alpha_V)$$

$$\alpha_C = \alpha_G + \alpha_V - \alpha_G\alpha_V$$

Após combinar cor e opacidade provenientes da voxelização dos polígonos e dos dados volumétricos, aplica-se um qualquer um dos algoritmos de *rendering* direto para gerar a imagem final da cena. Na Figura 5.1 utilizou-se o algoritmo *Ray Casting*.

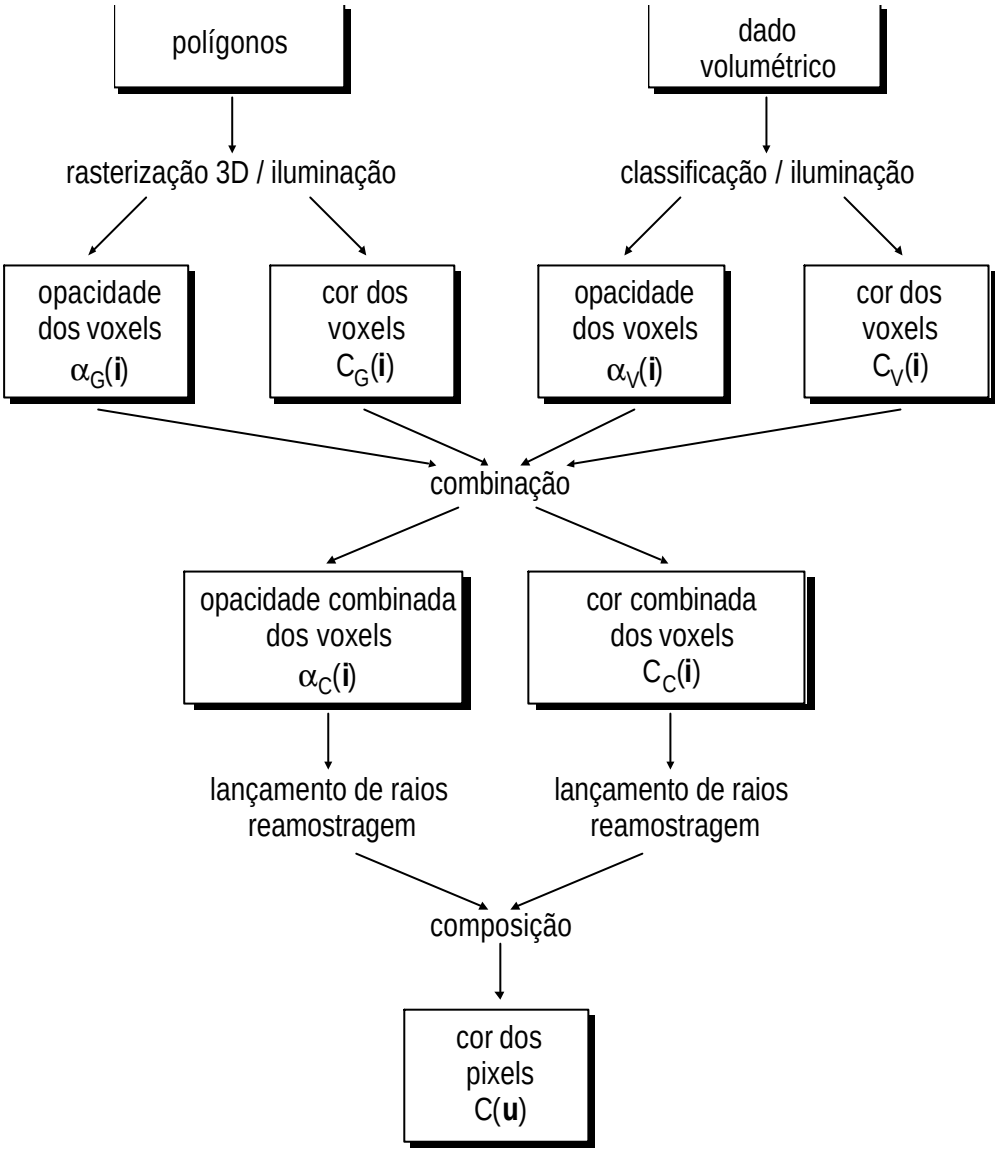


Figura 5.1: Conversão em uma representação comum volumétrica [Levoy, 90b]

5.2 Métodos Híbridos

Muitos dos algoritmos híbridos de visualização volumétrica são baseados no algoritmo *Ray Casting* [Levoy, 90a; Fruhauf, 91; Miyazawa e Koyamada, 92]. Basicamente, estes algoritmos calculam a interseção dos raios com o volume e polígonos em separado e, a seguir, combinam as interseções ao longo do raio.

Em [Kaufman e outros, 90] pode-se encontrar um algoritmo que utiliza dois *z-buffers* independentes, um para o volume e o outro para os objetos geométricos. Os dois *z-buffers* são combinados no final, gerando a imagem completa da cena. Entretanto,

como os *z-buffers* foram calculados de modo independente, este algoritmo não permite tratar volumes semi-transparentes.

[Tost e outros, 93] apresentam um algoritmo híbrido baseado na projeção dos *voxels* (*rendering* direto no espaço dos objetos) no plano de visualização. Este algoritmo utiliza uma *octree* para decompor a geometria, facilitando a paralelização. Além disso, permite que o volume e os objetos geométricos sejam transparentes.

Finalmente, [Schmidt e outros, 99] propõem uma extensão do algoritmo *Shear-Warp* capaz de tratar dados volumétricos e superfícies poligonais. Isto é obtido através da combinação do *Shear-Warp* (para tratar os dados volumétricos) com o *Z-buffer* (para tratar as superfícies poligonais).

5.2.1 Ray Casting

O algoritmo clássico de *Ray Casting* híbrido pode ser encontrado em [Levoy, 90a]. A descrição geral de seu funcionamento é apresentado na Figura 5.2. Raios paralelos são lançados a partir de cada *pixel* da tela. Para cada raio, é calculado um conjunto de cores e opacidades através da reamostragem do volume em intervalos regulares ao longo do raio, utilizando interpolação trilinear. Por outro lado, todas as interseções dos raios com os polígonos são calculadas independentemente e iluminadas utilizando o modelo Phong, gerando também um conjunto de cores e opacidades. Finalmente, todas as amostras são compostas a partir da mais próxima do observador até a mais distante, obtendo-se assim, a cor final de cada *pixel* na tela. Para isto, utiliza-se o método usual apresentado na seção 4.2.1.

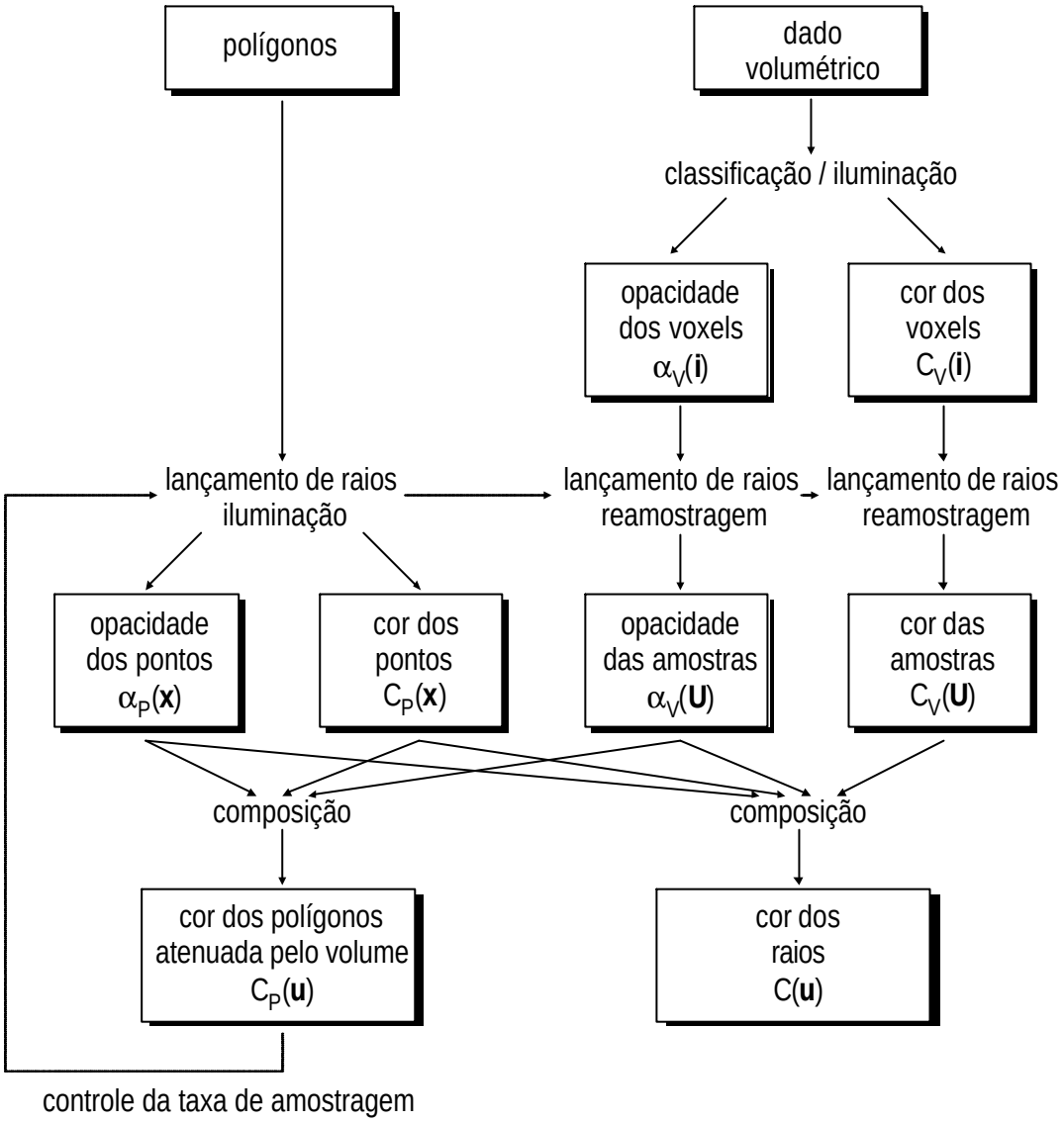


Figura 5.2: Ray Casting Híbrido [Levoy, 90a]

O problema encontrado neste algoritmo, e em todos baseados em amostragem, é a existência de *aliasing* caso não seja tomado alguns cuidados. Para que isto não ocorra, o autor utiliza uma técnica de super amostragem adaptativa:

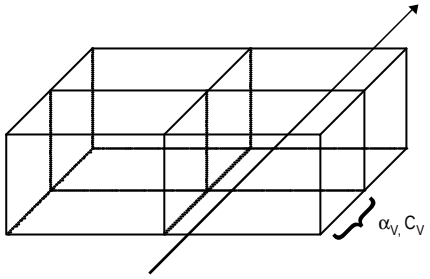
1. O plano de imagem é dividido em regiões adjacentes e um raio é lançado em cada um dos quatro cantos da região. Pela Figura 5.2, o algoritmo calcula a cor `C` final de cada raio (*pixel*) após a composição das cores e opacidades provenientes do volume e interseções com os polígonos;

2. Se a diferença entre as quatro cores resultados for inferior a um certo valor limite (escolhido pelo usuário), então não há necessidade de nova subdivisão e os *pixels* podem ser acesos na tela;
3. Caso contrário, um novo conjunto de quatro raios é lançado de forma análoga ao passo 1, entretanto, a cor final C_P dos *pixels* é calculada considerando-se apenas as contribuições dos polígonos, ou seja, a cor volume é desprezada ($C_V=0$), porém os raios são atenuados devido opacidade α_V do volume;
4. Se, nesse novo conjunto de quatro cores, a diferença não exceder o limite, então isso significa que a diferença de cor observada no passo 1 é devido ao volume, então não há necessidade de nova subdivisão e os *pixels* podem ser acesos na tela;
5. Caso contrário, a diferença de cor no passo 1 é devido aos polígonos, então deve-se subdividir a região considerada.

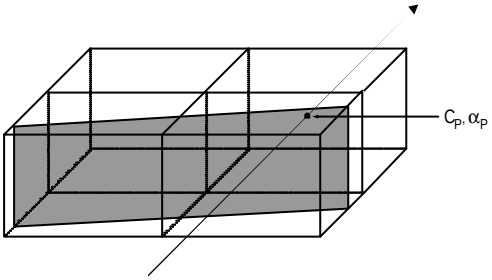
Outra questão considerada pelo autor é o correto cálculo da visibilidade entre os polígonos e o volume pois, devido a amostragem ao longo dos raios, os resultados podem ser aproximados. A Figura 5.3 ilustra esse problema. Cada paralelepípedo mostrado na Figura 5.3a corresponde a um pequeno pedaço do volume, cuja largura e altura correspondem ao espaçamento entre raios vizinhos e a profundidade corresponde ao espaçamento entre duas amostras consecutivas ao longo do raio. Os polígonos existentes na cena interceptam esses paralelepípedos, obscurecendo uma parte do volume do paralelepípedo e sendo obscurecido por outra parte. Dois casos diferentes são considerados:

1. O polígono intercepta apenas as faces laterais do paralelepípedo (Figura 5.3b). Nesse caso, obtem-se a solução exata (Figura 5.3c), subdividindo-se o paralelepípedo no ponto de interseção do raio com o polígono. De acordo com esse ponto, calcula-se a opacidade da parte anterior e posterior do paralelepípedo, em função da espessura de cada parte. A seguir, deve-se compor a parte anterior do paralelepípedo, depois o polígono e, por último a parte posterior do paralelepípedo. Este procedimento está descrito em [Levoy, 90a];
2. O polígono intercepta a face anterior ou posterior do paralelepípedo (Figura 5.3d). Nesse caso, o polígono contribuiu em mais de um paralelepípedo, apesar de interceptar o raio apenas uma vez. A solução exata para este caso requer um

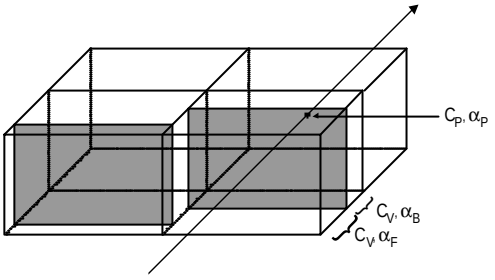
esforço computacional maior e uma possível aproximação pode ser obtida subdividindo-se o paralelepípedo (Figura 5.3d) e recaindo-se no caso anterior.



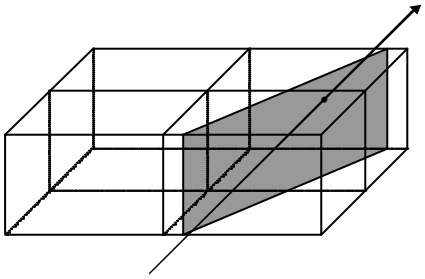
(a) subvolume e um raio de visão



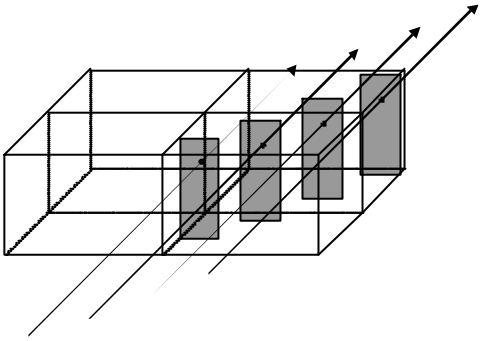
(b) Polígono intercepta apenas as laterais



(c) Solução exata de (b)



(c) Polígono intercepta na frente ou atrás



(d) Solução aproximada de (c)

Figura 5.3: Cálculo da visibilidade [Levoy, 90a]

5.2.2 Splatting

Uma boa adaptação do algoritmo de *Splatting* para tratar superfícies pode ser encontrado em [Tost e outros, 93]. Uma das motivações deste trabalho era, na época, a falta de um algoritmo de mapeamento direto (espaço dos objetos) capaz de visualizar corretamente volume e superfícies transparentes. Para atingir este objetivo, os autores classificam todas

as superfícies dentro do volume através da construção de uma *octree* do modelo de superfícies.

A *octree* é uma representação hierárquica baseada na decomposição recursiva do espaço tridimensional em cubos (octantes) que podem conter objetos ou partes dos objetos da cena (Figura 5.4). No caso deste trabalho, a *octree* armazena uma decomposição apenas dos objetos geométricos (superfícies e sólidos). A raiz desta estrutura representa toda a cena que a seguir é subdividida em oito cubos idênticos. De acordo com a presença ou não de objetos geométricos dentro dos cubos, estes podem sofrer novas subdivisões até um limite pré-estabelecido.

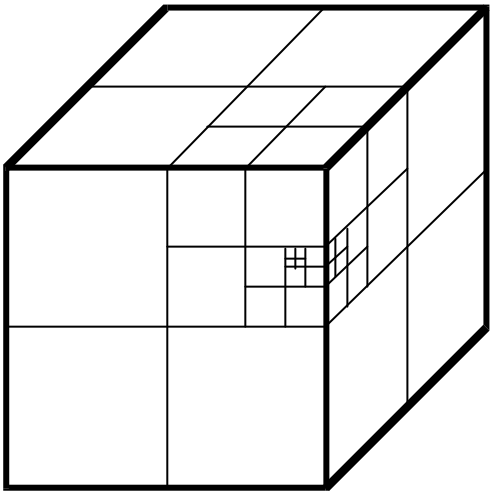


Figura 5.4: Estrutura de dados *Octree*

Em uma *octree* clássica, os nós das folhas podem ser de dois tipos diferentes: contém (tipo PRETO) ou não contém um objeto (tipo BRANCO). Existe também os nós não-terminais (tipo CINZA) que correspondem àqueles que necessitam de subdivisão. Entretanto, [Tost e outros, 93] utilizam uma variação desta estrutura, acrescentando mais um tipo de nó, chamado de FACE. Este nó é cortado por um único plano sólido (Figura 5.5). Os autores acreditam que esta estrutura, chamada de *Face Octree*, apresenta várias vantagens em relação a estrutura clássica. A primeira é ser muito mais compacta, necessitando um número muito menor de subdivisões, economizando, portanto, memória e tempo de processamento. A segunda vantagem é não voxelizar totalmente os objetos geométricos. Além disso, se a resolução da *octree* for compatível com a resolução do

modelo volumétrico, o erro cometido na aproximação da geometria pode ser comparável ao erro cometido na projeção dos *voxels*, reduzindo, assim, os efeitos da aproximação.

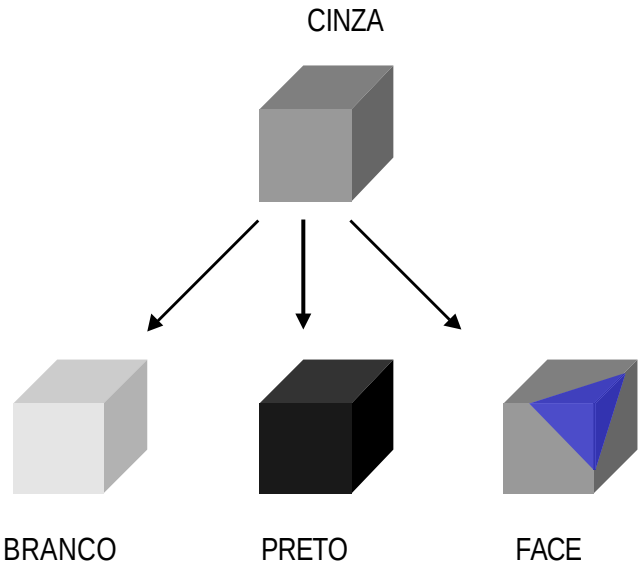


Figura 5.5: Estrutura de dados *Face Octree*

A construção da *octree* pode ser feita em uma etapa de pré-processamento, podendo, portanto, ser reutilizada caso a cena seja visualizada a partir de uma nova posição do observador. Entretanto, provavelmente devido a restrições de memória existentes na época, a *octree* era construída e percorrida durante a visualização da cena.

A idéia central do algoritmo é percorrer simultanemente a *octree* a partir de sua raiz (representando toda a cena) e o volume a partir do ponto mais distante do observador ao mais próximo (de trás para frente). A cada nó visitado, a geometria contida em seu interior é classificada e as seguintes situações podem ocorrer:

- O nó contém parcialmente um objeto geométrico. Neste caso, o nó é do tipo CINZA e requer uma nova subdivisão, até que seja encontrado um nó do tipo BRANCO, PRETO ou FACE ou o nó atingir um determinado tamanho mínimo (baseado no número de *pixels* de sua projeção ou tamanho do *voxel*);
- O nó não contém nenhum objeto geométrico. Neste caso, o nó é do tipo BRANCO e todos os seus *voxels* são projetados no plano de imagem (utilizando a técnica de *Splatting*) a partir do *voxel* mais distante ao mais próximo do observador;

- O nó contém algum objeto geométrico. Neste caso, o nó é do tipo PRETO e duas situações diferentes podem ocorrer: o objeto geométrico é vazio (só possui a fronteira definida) ou cheio (contém um sólido preenchido). Na primeira situação, a fronteira do objeto é exterior ao nó, portanto deve ser tradado como se fosse um nó do tipo BRANCO. Na outra situação, o nó é projetado de acordo com a cor e transparência do objeto;
- O nó contém uma face de um objeto geométrico. Nesse caso, o nó é do tipo FACE e todos os seus *voxels* são visitados de trás para frente. Se o *voxel* estiver do lado de dentro ou do lado de fora do objeto ele é projetado no plano de imagem como se fosse um nó do tipo PRETO ou BRANCO respectivamente. Entretanto, se o *voxel* estiver na fronteira do objeto, este é cerceado em relação ao *voxel* e projetado utilizando o modelo Phong de iluminação local. Observe que, neste caso, o volume próximo a fronteira do objeto é completamente ignorado. Isto pode ser resolvido subdividindo-se o *voxel*, muito semelhante ao algoritmo *Dividing Cubes* [Lorensen e Cline, 87].

5.2.3 Shear-Warp

Em [Schmidt e outros, 99] pode-se encontrar um estudo sobre a integração do algoritmo *Shear-Warp* e *Z-Buffer* resultando em um algoritmo híbrido. Uma das vantagens desse algoritmo é que a maioria das placas gráficas implementam o *Z-Buffer* em *hardware* com eficiência. Este trabalho tem a preocupação de reduzir ao máximo o efeito de *aliasing* introduzido durante o processo de amostragem das superfícies. Este problema ocorre sempre que a resolução do volume for baixa em comparação com a resolução da imagem final visualizada pelo usuário.

A idéia principal do algoritmo é mostrado na Figura 5.6. Tanto as fatias do volume como as superfícies poligonais sofrem a transformação de *shear*, levando a duas imagem intermediárias independentes que, a seguir, são compostas fatia a fatia. A seguir, a transformação de *warp* é aplicada na imagem composta levando a imagem final.

Para evitar os problemas de *aliasing*, as superfícies poligonais devem ser amostradas em uma resolução comparável com a resolução da imagem final. Para isso, a transformação de *warp* deve ser modificada de modo a mapear um *pixel* da imagem intermediária em um *pixel* na imagem final. Isso significa que as dimensões da imagem intermediária devem ser multiplicadas por dois fatores M e N, o que pode ser

interpretado como uma subdivisão de um *pixel* na imagem intermediária em MxN *sub-pixels*. Note, entretanto, que uma das principais vantagens do algoritmo *Shear-Warp* é realizar a etapa de composição na resolução da imagem intermediária, que é comparável às dimensões do dado volumétrico, e não da imagem final que, tipicamente, tem dimensões maiores.

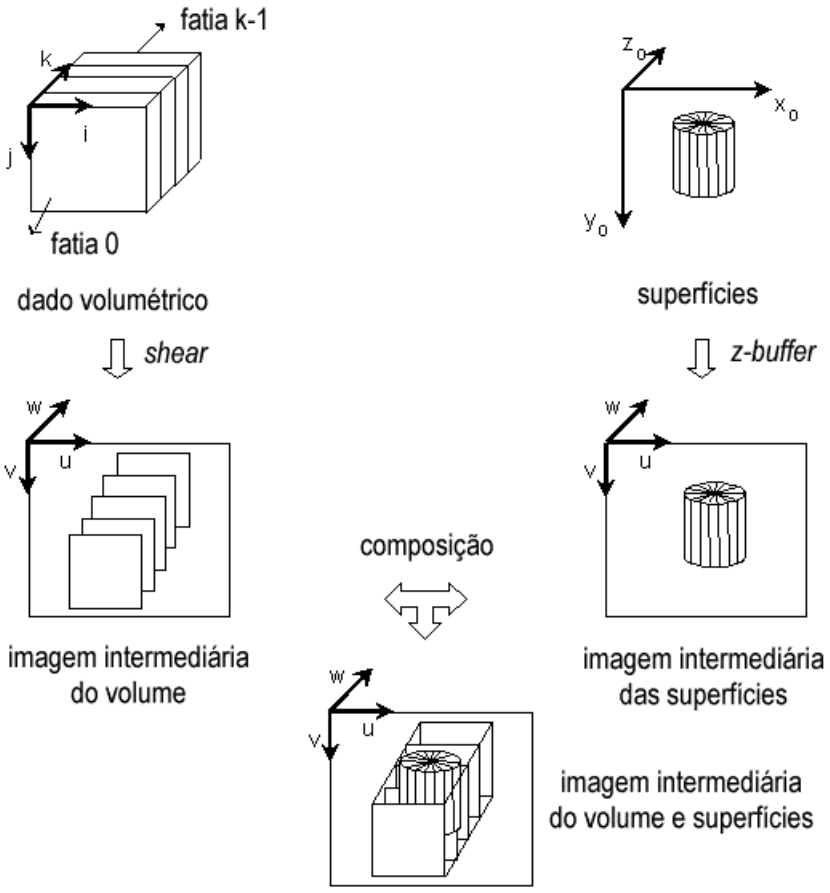


Figura 5.6: *Shear-Warp* Híbrido [Schmidt e outros, 99]

Os autores propõem três tipos diferentes de algoritmos para a integração do *Shear-Warp* e *Z-Buffer*. A diferença básica está na resolução utilizada para as imagens intermediárias. Baixa resolução significa a resolução normal da imagem intermediária do algoritmo *Shear-Warp* padrão. Alta resolução significa ampliar a imagem intermediária de forma que um *pixel* nesta imagem corresponda a um *pixel* na imagem final. Conforme comentado anteriormente, isso reduz o efeito de *aliasing* introduzido na discretização das superfícies. Deste modo, os algoritmos propostos podem ser classificados nos seguintes grupos:

- *Composição em baixa resolução*. Neste caso, a imagem intermediária do volume e das superfícies são construídas em baixa resolução, podendo ocasionar *alias* na imagem final, caso sua resolução seja maior que as imagens intermediárias;
- *Composição em alta resolução*. Neste caso, as duas imagens intermediárias são construídas em alta resolução. Esta técnica pode resultar em um algoritmo ineficiente, pois não há necessidade de compor toda a cena em alta resolução quando há contribuição apenas do volume (não há interferência de superfícies);
- *Composição em resolução dual*. Nessa situação, a imagem intermediária das superfícies é construída em alta resolução, porém a imagem intermediária do volume pode ser construída em baixa ou alta resolução, dependendo da influência de superfícies. Essa técnica procura evitar a subdivisão dos *pixels* quando não há necessidade.

Com relação ao último tipo de algoritmo (composição dual), os autores utilizam dois critérios diferentes para avaliar a necessidade de subdivisão dos *pixels* da imagem intermediária do volume:

- Footprint *das superfícies*. Nesse caso, todos os *pixels* sobre a projeção de qualquer superfície na imagem intermediária é subdividido;
- *Detecção de regiões de alta freqüência*. Nesse caso, é aplicado um filtro Laplaciano 3x3 sobre a imagem intermediária das superfícies produzindo uma nova imagem onde as regiões de alta freqüência (bordas e partes com iluminação irregular) possuem altas componentes RGB. Essas regiões são detectadas baseadas em um valor escalar limítrofe fornecido pelo usuário.

Dependendo do tipo de algoritmo utilizado, a etapa de composição pode variar um pouco. Entretanto, a construção da imagem intermediária das superfícies é baseada no algoritmo *Z-Buffer* padrão, que pode ser realizado 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 propriamente dita é feita fatia a fatia, junto com a construção da imagem intermediária do algoritmo *Shear-Warp* padrão. Isso é realizado a partir da fatia

mais próxima do observador até a mais distante. Após processar cada fatia, os algoritmos contabilizam a contribuição das superfícies, levando em consideração todas as superfícies existentes entre a fatia corrente e a seguinte. Para isto, uma lista auxiliar é construida ordenando os *pixels* da imagem intermediária das superfícies por profundidade. Com isso, não é necessário varrer todo o *z-buffer* ao ser processada cada fatia do volume.

Uma característica importante da integração do algoritmo *Shear-Warp* e *Z-Buffer* é a dificuldade de implementar transparência das superfícies. No *Z-Buffer* padrão só há informação da profundidade da superfície mais próxima em um determinado *pixel*. Isso significa que, durante a etapa de composição, somente a contruibuição de no máximo uma superfície pode ser considerada. Para isso, é necessário, além dos *buffers* de cor e profundidade, um *buffer* de opacidade. Isso pode ser facilmente implementado, porém só se consegue um nível de profundidade.

Para que a transparência das superfícies seja corretamente implementada, pode-se utilizar a estrutura Zlist-buffer proposta em [Zakaria e Saman, 99]. Nesse trabalho, 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. Entretanto, em muitas aplicação, um nível de transparência já pode ser suficiente. Em [Schmidt e outros, 99] foi implementado um algoritmo que permite até cinco níveis de transparência, utilizando uma estrutura semelhante ao Zlist-buffer.

6 CONCLUSÕES

Este trabalho incialmente estuda uma representação matemática unificada para dados volumétricos e geométricos. Conceitualmente, pode-se utilizar o mesmo modelo, baseado em funções do espaço ambiente, para representar esses dados.

A seguir, são estudads algumas técnicas de conversão de representação volumétrica em geométrica e vice-versa. A principal motivação para este estudo é que em muitas situações é necessário extrair superfícies relacionadas com o dado volumétrico. Isto confirma a necessidade de se utilizar um modelo híbrido, capaz de representar satisfatoriamente dados heterogêneos.

Depois são apresentados os principais conceitos de visualização volumétrica tradicional juntamente com os algoritmos básicos de *rendering* direto. Finalmente, são apresentados os problemas relacionados com visualização de dados heterogêneos. Algumas estratégias de adaptação dos algoritmos básicos são estudas em mais detalhes, buscando, assim, analisar as questões mais relevantes dos algoritmos híbridos de visualização volumétrica.

A principal motivação para esse estudo é, sem dúvida, as aplicações práticas da visualização de dados heterogêneos. Em muitas situações os dados volumétricos são estruturados e isso pode ser muito importante para auxiliar o usuário na visualização e interpretação dos dados. Na área médica, por exemplo, pode-se tirar partido da estrutura do dado oferecendo ao usuário ferramentas interativas mais eficientes pois, com uma representação explícita dos dados, é possível guiar o usuário durante a navegação, auxiliar o desenvolvimento e implantação de próteses, medir área e volume de tumores, auxiliar no planejamento cirúrgico etc.

Ainda existem muitos desafios para essa área. O primeiro deles é a característica exploratória da visualização volumétrica. Neste ponto, nota-se claramente a diferença entre fotorealismo, que está preocupado com a imagem final, e a visualização volumétrica, que está preocupada com a exploração do dado volumétrico. A visualização exploratória requer algoritmos eficientes e computadores bastante rápidos para permitir interatividade. Além disso, os dados volumétricos, por serem tipicamente grandes, podem

requerer muita memória para serem manipulados, agravando ainda mais o problema da interatividade.

Um outro importante desafio é o desenvolvimento de ferramentas de análise visando a identificação automática de estruturas dentro do volume. Neste caso, é muito importante considerar a existência de dados heterogênos. Uma aplicação prática é o reconhecimento automático de tumores em órgãos do corpo humano, existência de fraturas em ossos etc. Além disso, a própria visualização pode guiar o usuário durante a análise, permitindo, por exemplo, focar o estudo de uma fratura em uma peça mecânica em uma determinada região definida pelo usuário, baseado nos resultados obtidos durante a visualização.

7 REFERÊNCIAS

[Breen e outros, 88] Breen, D., Mauch, S., Whitaker, R., *3D Scan Conversion of CSG Models Into Distance Volumes*, Proceedings of the 1998 Symposium on Volume Visualization, ACM SIGGRAPH, October 1998, pp. 7-14, <http://waggle.gg.caltech.edu/~david/Abstracts/3D-scan-conv-abs.html>.

[Drebin e outros, 88] Drebin, R. A., Carpenter, L., Hanrahan, P., *Volume Rendering*, Computer Graphics, Volume 22, Number 4, pp. 65-74, 1988.

[Elvis, 92] Elvis, T. Tood, *A Survey of Algorithms for Volume Visualization*, Computer Graphics, Volume 26, Number 3, August 1992, pp 194-201.

[Fruhauf, 91] Fruhauf, M., *Combining Volume Rendering with Line and Surface Rendering*, Eurographics'91, 1991.

[Gerhardt e outros, 98] Gerhardt, A., Paiva, A., Schmidt, A.E., Martha, L.F.; Carvalho, P.C., Gattass, M., *Aspects of 3-D Seismic Data Volume Rendering*, Proceedings of the GOCAD ENSG Conference – 3D Modeling of Natural Objects: A Challenge for the 2000's, Nancy, França, 04 a 05 de junho de 1998.

[Gomes e Velho, 92] Gomes, J. M., Velho, L., *Implicit Objetcts in Computer Graphics*, Monografia de Matemática nº 53, Instituto de Matemática Pura e Aplicada, 1992.

[Gomes e Velho, 97] Gomes, J. M., Velho, L., *Image Processing for Computer Graphics*, Springer-Verlag, 1997.

[Gouraud, 71] Gouraud, H., *Continuos Shading of Curved Surfaces*, IEEE Transactions of Computers, Vol. 20, No. 6, pp. 623-629, 1971.

[Kaufman, 87] Kaufman, A. E., *Efficient algorithms for 3D scan-conversion of parametric curves, surfaces and volumes*, Computer Graphics (SIGGRAPH'87 Proc) 21(4), 171-179, 1987.

[Kaufman e outros, 90] Kaufman, A., Yagel, R., Cohen. D., *Intermixing Surface and Volume RenderingI*, 3D Imaging in Medicine: Algoritms, Systems and Applications, K.H. Hohne, H. Fuchs, S.M. Pizer, Springer-Verlag, 217-228, 1990.

[Kaufman e Sobierajski, 94] Kaufman, A. E., Sobierajski, L. M., *Continuum Volume Display*. Computer Visualization: Graphics Techniques for Scientific and Engineering Analysis, Chapter 6, Edited by Richard S. Gallagher, pp171-202, 1994.

[Kaufman, 97] Kaufman, A. E., *Volume Visualization: Principles and Advances*. SIGGRAPH'97 Course Notes, 1997.

[Lacroute, 95] Lacroute, P., *Fast Volume Rendering Using a Shear-Warp Factorization of theViewing Transformation*", Ph.D. Thesis, Stanford University, 1995.

[Levoy, 88] Levoy, M., *Display of Surfaces from Volume Data*", IEEE Computer Graphics and Applications, Vol. 5, No. 3, pp 29-37, 1988.

[Levoy, 90a] Levoy, M., *A Hybrid Ray Tracer for Rendering Polygon and Volume Data,* IEEE Computer Graphics and Applications, Vol. 10, No. 2, March, 1990, pp. 33-40.

[Levoy, 90b] Levoy, M., *Ray Tracing of Volume Data,* SIGGRAPH'90 Course Notes, Volume Visualization Algorithms and Archtectures, August, 1990, pp. 120-147.

[Levoy, 90c] Levoy, M., *Efficient Ray Tracing of Volume Data,* Computer Graphics (Proceedings of SIGGRAPH'90), pp. 157-167, 1990.

[Lorensen e Cline, 87] Lorensen, W. E., Cline, H. E., *Marching Cubes: A High Resolution 3D Surface Construction Algorithm*, Computer Graphics (Proceedings of SIGGRAPH'87), Vol. 21, No. 4, pp 163-169, 1987.

[Miyazawa e Koyamada, 92] Miyazawa, T., Koyamada, K., *A High-speed Integrated Renderer for Interpreting Multiple 3D Volume Data,* The Journal of Visualization and Computer Animation, 3, 65-83, 1992.

[Novins e outros, 90] Novins, K. L., Sillion, F., Greenberg, D. P., *An efficient method for volume rendering using perspective projection*. Computer Graphics, Vol. 24, No. 5, pp 95-102.

[Paiva e outros, 99] Paiva, A. C., Seixas, R. B., Gattass, M., *Introdução à Visualização Cientifica*, Monografia em Ciência da Computação, nº 3/99, Departamento de Informática, PUC-Rio, 1999.

[Pavlidis, 78] Pavilidis, T., *Filling algorithms for raster graphics*, Computer Graphics (SIGGRAPG'78 Proc.) 12(3), 161-166, 1978.

[Phong, 75] Phong, B. T., *Illumination for Computer Generated Pictures*, Communications of ACM, Vol. 18, No. 6, pp. 311-317, 1975.

[Porter e Duff, 84] Porter, T., Duff, T., *Compositing Digital Images*, Computer Graphics, Vol. 18, No. 3., July, 1984.

[Robb e outros, 99] Robb, R. A., Camp, J. J., Hanson, D. P., *Computer Aided Surgery and Treatmeny Planning at the Mayo Clinic*, Biomedical Imaging Resource, Mayo Foundation/Clinic, Rochester, MN, 1999, http://www.mayo.edu/bir/reprints/CAS.html.

[Schmidth e outros, 99] Schmidth, A. E., Gattass, M., Carvalho, P. C., *Combined 3D Visualization of Volume Data and Polygonal Surfaces Using a Shear-Warp Algoritm*, Monografia em Ciência da Computação, nº 5/99, Departamento de Informática, PUC-Rio, 1999.

[Speray e outros, 90] Speray, D., Kennon, S., *Volume Probes: Interactive Data Exploration on Arbitrary Grids*, Computer Graphics, Vol. 24, No. 5, pp 1-12, 1990.

[Tost e outros, 93] Tosti, Dani, Puig, A., Navazo, I., *Visualization of mixed scenes based on volume and surfaces*. Fourth Eurographics Workshop on Rendering, pp. 281-293, 1993.

[Upson e Keeler, 88] Upson, C., Keeler, M., *V-Buffer – Visible Volume Rendering*, Computer Graphics, Vol. 22, No. 4, August, 1988.

[Velho e outros, 95] Velho, L., Terzopoulos, D., Gomes, J., *Constructing Implict Shape Models from Boundary Data*, Computer Vision and Image Processing, Vol. 57, No. 3, pp. 220-234, May, 1995.

[Wang e Kaufman, 93] Wang, S., Kaufman, A., *Volume Sampled Voxelization of Geometric Primitives*, Visualization ’93 Proceedings, pp 78-84, San Jose, CA, October, 1993.

[Westover, 90] Westover, L., *Footprint Evaluation for Volume Rendering*, Computer Graphics (Proc. SIGGRAPH), 24 (4), pp. 144-153, August, 1990.

[Zakaria e Saman, 99] Zakaria, M. N., Saman, M. Y., *Hybrid Shear-Warp Rendering*, Proceedings of the ICVC, Goa, India (1999).