Visualização de Imagens Médicas Volumétricas

Leonardo Marques Rocha

11 de julho de 2004

Capítulo 1

Introdução

A visualização volumétrica tem sido utilizada em várias áreas tais como microscopia, modelagem molecular, astrofísica, geofísica, química e engenharia. Na medicina, a visualização pode auxiliar no diagnóstico por imagens, no planejamento e acompanhamento de cirurgias, em tratamentos por radiação, no estudo da cinemática de articulações, na caracterização de doenças que afetam a anatomia e a fisiologia de determinadas estruturas, e na educação médica [1]. Nessas aplicações, a principal meta é auxiliar os especialistas na compreensão de fenômenos físicos complexos do corpo humano tendo como base, dados relacionados com parâmetros físicos, tais como, medidas de densidade, velocidade, carga eletrostática e entropia. O enfoque desta monografia é a visualização de volumes aplicada à medicina.

Pesquisadores ao longo das últimas duas décadas têm se empenhado em desenvolver técnicas de processamento, visualização e análise de dados biomédicos 3D. Graças ao aumento da velocidade de processamento e capacidade de memória dos computadores, e ao surgimento de novas modalidades de aquisição de dados - tomografia por Raios-X (CT e Spiral CT), tomografia por emissão de pósitron (PET), tomografia por emissão de fóton (SPECT), ressonância magnética (MRI), ressonância magnética funcional (fMRI), ultrasom, microscopia confocal -um rápido crescimento no desenvolvimento de *hardware* e *software* para implementação dessas técnicas tem sido possível.

Estes equipamentos adquirem dados na forma de imagens de fatias paralelas e ortogonais a um dos eixos principais do paciente (e.g. cortes transversais, sagitais e coronais). O empilhamento destas imagens, mantendo o espaçamento original entre elas, pode ser idealizado de forma que cada pixel representa o volume de um pequeno paralelepípedo no espaço 3D, denominado *v*oxel. Este volume de dados é denominado *c*ena e representa amostras

em uma região 3D do corpo do paciente. Cada voxel tem associado um número inteiro proporcional ao tom de cinza do pixel na imagem correspondente. Cada um desses valores representa a integração de uma propriedade física que está sendo mensurada no interior da cena. Em uma cena os objetos de interesse para visualização são órgãos ou outras estruturas do corpo associadas ao problema em estudo.

1.1 Visualização em Medicina

Existem três importantes etapas em visualização de volumes na medicina - classificação, representação, e *rendering*. *Classificação* é processo de atribuir aos voxels opacidades proporcionais ao grau de interesse nesses voxels para visualização. O maior desafio na classificação é identificar que voxels pertencem a objetos distintos, que é, em essência, um problema de segmentação [2]. *Representação* consiste em definir um modelo geométrico para o objeto classificado e uma estrutura de dados para armazenar o modelo junto com um conjunto de atributos para visualização. *Rendering* é o processo de simulação da propagação dos raios de luz através da cena e da determinação da luz que alcança o observador para criar diferentes vistas dos objetos selecionados.

Abordagens de visualização podem ser classificadas em dois grupos - *rendering* de superfície e *rendering* de volume. Em *rendering* de superfície, os voxels são classificados como opacos ou transparentes [3, 4, 5, 6]. Como conseqüência, um modelo geométrico da superfície dos objetos pode ser criado. Os métodos podem ser posteriormente divididos em duas classes de modelos geométricos - poligonal e digital. Métodos digitais representam a superfície dos objetos como um conjunto de primitivas - voxel e faces de voxel - que estão diretamente associados com o sistema de coordenadas da cena [5, 6]. Em modelos poligonais, um conjunto de polígonos - mais comumente triângulos - são utilizados para representar a superfície [3, 4]. Em *rendering* de volume, o processo de classificação é nebuloso onde voxels têm um grau contínuo de opacidade que varia de transparente a opaco [6, 7, 8, 9]. O modelo geométrico dos objetos é digital (um conjunto de voxels), e no caso de classificação binária, *rendering* de volume é equivalente ao *rendering* de superfície digital, o que torna esta abordagem interessante, simples e flexível para visualização.

1.2 Objetivo

Esta monografia faz uma revisão dos principais conceitos sobre visualização de volumes.

Capítulo 2

Visualização de volumes

Este capítulo apresenta os conceitos básicos em visualização de volumes, provenientes de áreas de processamento de imagens [2] e computação gráfica [15]. Eles são organizados em cinco etapas principais de um sistema de visualização: aquisição, pré-processamento, classificação, representação e *rendering*. A aquisição consiste do processo de captura, digitalização e armazenamento de propriedades físicas na forma de uma seqüencia de imagens digitais, referida como cena. O pré-processamento utiliza operações de processamento de imagens para preparar a cena para classificação. A classificação associa um valor de opacidade aos voxels da cena proporcional ao grau de interesse para visualização. A representação define um modelo para os objetos classificados na cena e uma estrutura de dados para armazenar o modelo e as informações necessárias para visualização. O *rendering* simula a propagação de luz na cena para gerar as diferentes vistas dos objetos selecionados.

2.1 Aquisição de Imagens

A *aquisição* dos dados volumétricos é feita por um sensor para imageamento com capacidade de digitalização de sinais. Os dados volumétricos são adquiridos por tomógrafos em forma de imagens de cortes (ou fatias) paralelas ao longo de um dos eixos do paciente. Cada imagem está associada a uma localização $k \in \mathcal{Z}$ no eixo z com uma espessura e que pode variar de 1mm a 15mm, dependendo da resolução necessária à aplicação. Estas imagens têm tamanhos [1] típicos de 256×256 e 512×512 pixels, onde cada pixel tem a profundidade de 12 bits. Os espaçamentos entre fatias adjacentes são normalmente maiores que os espaçamentos entre pixels adjacentes da mesma

[1] A nova geração de tomógrafos vai produzir imagens de tamanhos 1024×1024 pixels

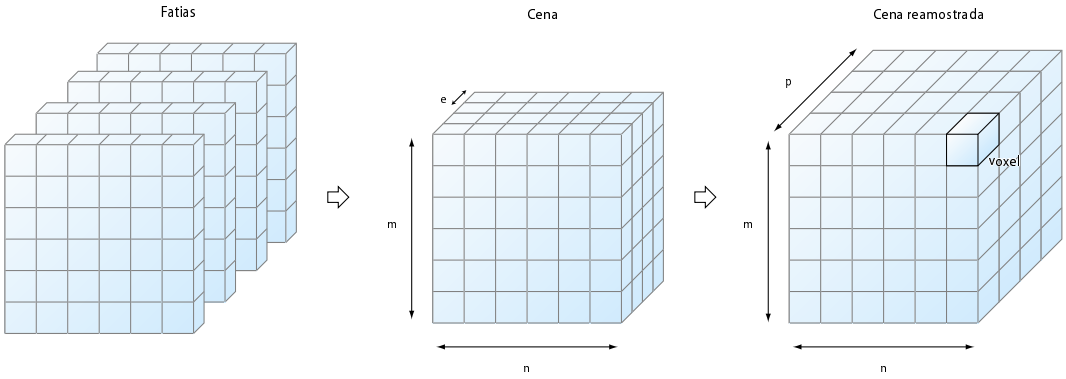


Figura 2.1: Imagens de fatias paralelas formam uma grade retangular finita V de voxels denominada cena.

fatia. As imagens são agrupadas na ordem crescente de sua posição k em uma única estrutura, chamada de cena, cujo elemento básico de construção é chamado de voxel (*volume element*). No caso em que a amostragem é variável ao longo do eixo z, o espaçamento entre fatias adjacentes torna-se variável, o que permite que determinadas regiões fiquem mais detalhadas em detrimento do detalhe de outras regiões.

Matematicamente, uma *cena* é um par (V, f) consistindo de uma grade retangular finita V de voxels (pontos em $\mathcal{Z}^3$), e um mapeamento f que designa para cada voxel v em V um valor de intensidade $f(v)$ em Z. Uma cena é comumente armazenada como uma matriz 3D de voxels, onde cada voxel representa um pequeno paralelepípedo em $\Re^3$. Uma cena pode conter um ou mais objetos de interesse para visualização.Um *objeto* em V é um conjunto O de voxels formando um ou mais componentes conexos em $\mathcal{Z}^3$. Cada objeto possui uma *borda* $B \subseteq O$, definida pelo conjunto de voxels formado por todos os voxels $v_b \in O$, onde a vizinhança de v_b intercepta O e o complemento de O.

2.2 Pré-processamento

O pré-processamento pode ser entendido como qualquer operação de processamento de imagens que mapeia um cena C_1 em outra cena C_2 visando aumentar as chances de sucesso na classificação e/ou extrair informação pa-

ra visualização. Estas operações são: filtragem, segmentação, volume de interesse, transformações radiométricas e reamostragem.

2.2.1 Filtragem

O uso de máscaras espaciais para processamento de imagens é usualmente chamado de *filtragem espacial* (em contrapartida à expressão *filtragem no domínio da freqüência* usando a transformada de Fourier). Nesta seção são considerados filtros lineares e não-lineares para filtragem de imagens [2].

Seja Γ uma filtragem que transforma uma cena C_1 em outra cena C_2. Γ é um filtro linear se:

$$\Gamma[aC_1 + bC_2] = a\Gamma[C_1] + b\Gamma[C_2] \tag{2.1}$$

para cenas C_1 e C_2 e a, b constantes quaisquer.

Os denominados filtros *passa-baixas* atenuam ou eliminam os componentes de alta-freqüência no domínio de Fourier enquanto deixam as freqüências baixas inalteradas. Filtros passa-baixas são normalmente usados para reduzir ruído de quantização e suavizar bordas para visualização através da geração de normais. De modo similar, filtros *passa-altas* atenuam ou eliminam os componentes de baixa-freqüência. O efeito resultante da filtragem passa-altas é um aparente realce das bordas e outros detalhes finos. Este tipo de filtro é útil no cálculo de vetores normais para visualização e na localização das regiões de bordas para classificação. Um terceiro tipo de filtragem, denominado filtragem *passa-banda*, remove regiões selecionadas de freqüências entre altas e baixas freqüências. Esses filtros são usados para restauração de cenas e raramente são interessantes para realce de cenas. Independentemente do tipo filtro linear usado, entretanto, a abordagem clássica consiste em somar os produtos entre os coeficientes da máscara e as intensidades dos voxels sob a máscara numa posição específica da cena.

Filtros espaciais não-lineares baseiam-se diretamente nos valores dos voxels na vizinhança considerada, não utilizando coeficientes como filtros lineares. Exemplos são os filtros mediana para eliminar ruído impulsivo e filtros morfológicos. Filtros morfológicos constituem alternativas à filtragem não-linear e são caracterizados por uma transformação Γ crescente e idempotente [16]. Neste sentido, a erosão e a dilatação, por não serem idempotentes, não constituem filtros morfológicos. Por sua vez, a abertura e o fechamento são exemplos típicos de operadores de filtragem, cuja composição também define filtros. A filtragem morfológica é muito útil também no pós-processamento de imagens, onde filtros de fechamento melhoram a qualidade da imagem gerada por *rendering*, removendo artefatos.

2.2.2 Segmentação

Um dos principais desafios em visualização de volumes é identificar quais voxels pertencem a quais objetos na cena. Esta operação é chamada segmentação. A segmentação subdivide uma cena em suas partes ou objetos constituintes. O nível até o qual essa subdivisão deve ser realizada depende do problema a ser resolvido [2, 1].

Segmentação tem duas tarefas relacionadas — reconhecimento e delineação. *Reconhecimento* é a tarefa, em alto nível, de determinar a posição aproximada do objeto na cena. *Delineação* é a tarefa, em baixo nível, de determinar a extensão espacial do objeto. Na maioria das tarefas de reconhecimento, operadores humanos treinados superam qualquer algoritmo de computador. Por outro lado, algoritmos de computador delineiam objetos de forma mais precisa, eficiente e exata que o ser humano.

Abordagens para reconhecimento podem ser classificadas em dois grupos: automática e assistida pelo usuário. Em métodos automáticos, uma entre duas abordagens é escolhida: *baseada no conhecimento* [17, 18, 19, 20] ou *baseada em atlas* [21, 22, 23, 24, 25, 26, 27]. A primeira abordagem usa técnicas de inteligência artificial para representar o conhecimento de objetos na região de interesse da cena. Usualmente, a delineação dos objetos é feita primeiro em uma fase de treinamento para depois serem formadas e testadas as hipóteses. Essas operações formam um laço que pode ser executado diversas vezes antes que uma solução aceitável seja encontrada. A segunda abordagem utiliza um atlas já construído que representa as relações geométricas e topológicas dos objetos. Conseqüentemente, algumas entidades geométricas, como pontos, contornos, planos, identificados na cena de entrada (passo de delineação) para uma mesma região do corpo são mapeados para casar com as entidades homólogas no atlas.

Em métodos de reconhecimento automático, a questão levantada é o que fazer em caso de falhas. Métodos completamente automáticos que são a prova de falhas e que têm funcionado corretamente de forma rotineira em um grande número de testes ainda não foram encontrados. A premissa em métodos assistidos pelo usuário é que uma simples ajuda de um operador é suficiente para o reconhecimento.

Abordagens para delineação, por outro lado, foram estudadas mais extensivamente que aquelas para reconhecimento. De fato, a delineação por si só é usualmente considerada um problema de segmentação. Isto é, qualquer que seja a saída do método de delineação, esta é considerada uma representação do objeto de interesse. Duas classes de métodos existem para delineação: *baseada em borda* [28, 29], em que a saída representa a borda

do objeto, e *baseada em região* [30, 31], em que a saída representa a região ocupada pelo objeto. Em ambos os grupos, a saída pode ser *binária* ou *nebulosa*. No caso binário, a saída é um conjunto de voxels, onde cada voxel tem um valor de pertinência ao objeto que é 0 ou 1. No caso nebuloso, a saída é um conjunto, onde um grau de pertinência ao objeto, no intervalo $[0, 1]$, é associado a cada voxel.

Considerando as duas estratégias de reconhecimento — automática e assistida pelo usuário — com as quatro abordagens de delineação, oito classes de estratégias de segmentação são possíveis.

2.2.3 Volume de Interesse

Geralmente o objeto de estudo ocupa apenas uma pequena porção do domínio da cena. A operação de *volume de interesse* (VOI) permite criar outra cena cujo domínio é um paralelepípedo que engloba a informação sobre todos os aspectos de interesse do objeto de estudo com o mínimo possível de objetos irrelevantes. O principal propósito da operação de VOI é minimizar o espaço de armazenamento necessário para o processamento futuro da cena. Como exemplo, uma cena $512 \times 512 \times 64$ contém mais de 16,5 milhões de voxels. Se a região ocupada pelo objeto de interesse é $200 \times 200 \times 64$, então o número de voxels — aproximadamente 2,5 milhões, — é cerca de sete vezes menos que a cena inteira. Outra razão para a operação de VOI pode ser a exclusão de uma parte do objeto para revelar o seu interior na visualização.

2.2.4 Transformações radiométricas

As transformações radiométricas independem da localização espacial dos voxels na cena e podem, de maneira geral, ser representadas por uma operação do tipo *look-up table* (LUT) que transforma o nível de cinza g_1 em um nível de cinza g_2. Seja G_1 a escala de cinza da cena de entrada C_1: $G_1 = [0, 1, \ldots, N_1]$ e G_2 a escala de cinza da cena resultante C_2: $G_2 = [0, 1, \ldots, N_2]$. Uma LUT é uma operação $r(G_1) \mapsto r(G_2)$ tal que

$$\forall g_1 \in G_2, \exists g_2 \in G_2, g_2 = r(g_1) \tag{2.2}$$

São exemplos de função radiométrica: *stretching*, normalização, limiarização e intensidade de interesse. O objetivo do *stretching* é realçar o contraste entre dois pontos na escala de cinza G_1. A limiarização é utilizada com freqüência no processo de segmentação, onde G_1 é mapeado em intensidades binárias (preto e branco). A operação de intensidade de interesse (IOI) tem o propósito de diminuir o espaço de armazenamento necessário

para cada voxel através da redução do domínio de f_1 para f_2, dentro de um intervalo de intensidades cuja representação computacional necessária é menor que em f_1. Por exemplo, valores de intensidade em um intervalo de 256 níveis de cinza podem ser representados por 8 bits ao invés de 12 bits da cena original.

2.2.5 Reamostragem

O objetivo da reamostragem é converter uma cena C_1 em outra cena C_2 com um nível especificado de discretização isotrópica. Isto é, uma cena onde todos os voxels de uma cena têm tamanhos iguais e então cada voxel pode ser identificado por uma tripla (x, y, z) em $\mathcal{Z}^3$ de coordenadas de seu centro. Esta operação é fundamental para evitar distorções na imagem visualizada.

Técnicas de reamostragem podem ser divididas em duas categorias: *baseada na cena* e *baseada no objeto* [1]. Em métodos baseados na cena, os valores de intensidade da cena reamostrada são determinados diretamente dos valores de intensidade da cena de entrada. Em métodos baseados no objeto, alguma informação do objeto é extraída da cena é usada no processo de reamostragem.

2.2.5.1 Métodos de reamostragem baseados na cena

Supondo que v' é um voxel na cena de saída C_2 e que v_i, $i = 1, 2, \ldots, 8$ são os voxels mais próximos na cena de entrada C_1, conforme ilustra a Figura 2.2. A forma mais simples de reamostragem é contribuir a v', a intensidade do voxel mais próximo entre os voxels v_i, $i = 1, 2, \ldots, 8$. Outra forma, muito comum em visualização de volumes, é a reamostragem linear, que assume que as intensidades dos voxels (g_1) variam linearmente ao longo das direções x, y e z, conforme a Equação 2.3:

$$g_2(v') = g_1(v_n) + [\frac{g_1(v_{n+1}) - g_1(v_n)}{d_{n+1} + d_n}]d_n. \tag{2.3}$$

Para determinar a intensidade em qualquer ponto c' (que representa o centro de um novo voxel v'), são encontrados oito voxels $v_1, v_2, \ldots, v_8$ em C_1 tais que seus centros $c_1, c_2, \ldots, c_8$ estão mais próximos de c'. O cálculo da intensidade em c' consiste de sete cálculos de reamostragem (Equação 2.3): a intensidade em r_1 é calculada através das intensidades de c_1 e c_2 e a intensidade em r_2 é calculada através das intensidades de c_5 e c_6. As intensidades de r_1 e r_2 são utilizadas no cálculo de reamostragem de r_3. A intensidade

em r_4 é calculada de forma similar, utilizando três cálculos. Finalmente, $g_2(v')$ é calculado com base nas intensidades de r_3 e r_4.

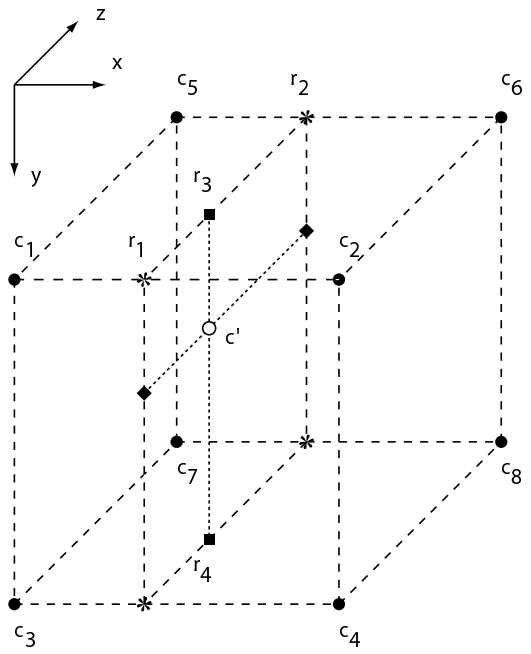


Figura 2.2: Reamostragem trilinear: $c_1, c_2, \ldots, c_8$ são os centros dos oito voxels que estão mais próximos de c', o centro de v'.

2.2.5.1.1 Refatiamento A transformação de refatiamento pode ser definida como uma operação de reamostragem feita na espaço de coordenadas da cena, para visualizar uma ou mais fatias em qualquer direção arbitrária, que não necessariamente x, y ou z. Esta transformação também permite que medidas quantitativas de distância e área possam ser feitas sobre as novas fatias.

A orientação de uma nova fatia pode ser ortogonal, oblíqua ou curva em relação ao espaço da cena de entrada. Sua localização, resolução e espessura também são parâmetros arbitrários. Portanto, o centro e o volume de um voxel nem sempre coincidem com o centro e o volume de um voxel do espaço original. Neste caso, torna-se necessária a utilização de técnicas de reamostragem para estimar a intensidade dos novos voxels. Os parâmetros de localização e orientação do refatiamento são especificados pelo usuário através de um plano ou uma linha.

Uma grande vantagem do refatiamento é que o paciente não precisa se submeter a um novo exame para que fatias em outras orientações sejam obtidas. No caso das tomografias, evita que o paciente receba uma dose maior de radiação. Em outros casos, a orientação desejada para as fatias é algumas vezes impossível de ser obtida pelo equipamento tomográfico.

2.2.5.2 Métodos de reamostragem baseados no objeto

O primeiro método baseado no objeto que aparece na literatura é conhecido como *reamostragem baseada na forma*, desenvolvido para cenas de entrada binárias [32]. Estas cenas são resultantes de um processo de segmentação, onde voxels com valor 1 representam o objeto de interesse e voxels com valor 0 representam o fundo. A reamostragem baseada na forma usa uma cena binária como entrada e gera outra cena binária para nível específico de discretização. O método consiste da conversão da cena de entrada binária C_b na cena C_g, ambas com mesmo domínio, pela designação, para cada voxel v em C_g, de um número que representa a menor distância entre v e a borda (mapa de distância Euclidiana) em C_b entre regiões do objeto e regiões na mesma fatia que contém v. O número é positivo se v tem uma intensidade 1 em C_b, caso contrário o número é negativo. A cena C_g é reamostrada usando regras de reamostragem trilinear ou qualquer outra regra de reamostragem para criar uma cena C'_g. Como os números associados com os voxels em C'_g representam as distâncias da borda, a cena reamostrada C'_b é criada pela designação da intensidade 1 se o voxel correspondente em C'_g tem valor positivo. Todos os outros voxels são designados com intensidade 0. Várias extensões deste método foram desenvolvidas [33, 34, 35].

2.3 Classificação

No modelo conceitual utilizado em *rendering* de volume, uma cena é considerada um bloco colorido e translúcido, com valores de cor e opacidade diferentes para cada região. A simulação da incidência de luz neste bloco permite que, através de sua reflexão, diferentes tipos de tecidos possam ser visualizados pelo observador. O processo de classificação atribui aos tecidos opacidades proporcionais ao grau de interesse nesses tecidos para visualização [36]. Na prática, os voxels de interesse pertencem às superfícies entre os tecidos e a localização desses voxels é um problema de segmentação. Outra forma de resolver o problema é segmentar os objetos e depois atribuir opacidades aos voxels da superfície e/ou vizinhança da superfície desses objetos.

Matematicamente, *classificação* é o processo de transformação de uma cena $C_1 = (V, f)$ em outra cena $C_2 = (V', \alpha)$ pela associação de um valor de opacidade $\alpha(v)$ dentro do intervalo $[0, 1]$ para cada voxel v em V'. Tal classificação é também chamada classificação nebulosa. A classificação binária é um caso particular onde os valores de opacidade $\alpha(v)$ são 0 ou 1. No caso de múltiplos objetos, a classificação pode associar um par ordenado (α, λ) para cada voxel da cena, onde α é o valor de opacidade no intervalo [0,1] e λ é um rótulo no conjunto $\{0, 1, \ldots, n-1\}$ que identifica o objeto. No caso da classificação binária, valores $\alpha \in \{0, 1\}$ e $\lambda \in \{0, 1, \ldots, n-1\}$ são associados.

A função que faz o mapeamento de opacidades para os voxels da cena é associada a uma ou mais características da imagem. Surgem dificuldades no processo de classificação quando esta característica da imagem não é capaz de dissociar dois objetos na cena. Uma classificação baseada apenas na intensidade dos voxels, por exemplo, deve apresentar estruturas em intervalos de intensidade separados. Este tipo de classificação apresenta bom resultado, como mostra a função de classificação de uma cena para pele e osso na figura 2.3. Infelizmente, na maioria das situações isto não ocorre, tornando o problema mais complexo.

Pesquisadores da Pixar, em 1985, foram os primeiros a utilizar técnicas de classificação de forma mais sofisticada em dados médicos 3D. Uma técnica utilizada, descrita em termos gerais por Smith [37] e apresentada em detalhes por Drebin et al. [36], consiste em estimar a fração de cada material no interior dos voxels e usar esta fração para calcular uma cor e uma opacidade parcial para cada voxel. O método de Levoy [38] é semelhante, mas calcula as cores e opacidades diretamente do valor escalar de cada voxel e na magnitude de gradiente. Quando o valor escalar escolhido é a intensidade dos voxels na cena, a classificação varia de acordo com os valores resultantes da multiplicação ponto a ponto das curvas de classificação de intensidade e magnitude de gradiente da cena, como mostra a Figura 2.3.

2.4 Representação

Os objetos classificados podem ser representados de diversas formas como descrito no Capítulo 1. Assumindo a representação digital, existem várias formas eficientes de armazenar os voxels de uma cena classificada. Um exemplo é a representação *octree*. Esta representação resulta de repetidas subdivisões do cena em subcenas até que todas as subcenas tenham apenas voxels pertencentes ao objeto ou não. A estrutura final é uma árvore, onde

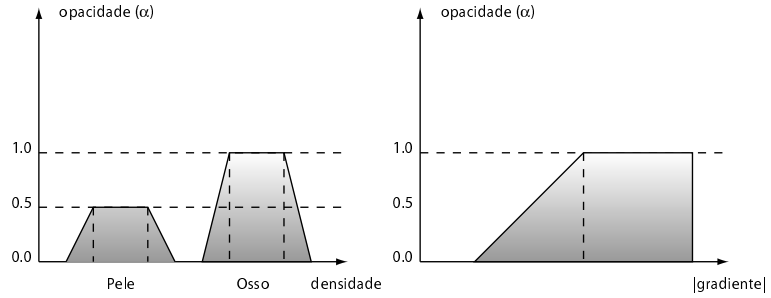


Figura 2.3: Função de classificação para pele e osso baseada na intensidade e na magnitude do gradiente.

cada nó tem oito ramos. Quando a cena tem mais de um objeto de interesse, a representação por *octree* é feita para cada objeto isoladamente. Apesar do acesso aos voxels não ser direto, operações simples de conjuntos, como união, intersecção e diferença, e transformações geométricas podem ser feitas acessando cada nó da árvore no máximo uma vez. Além disto, com exceção das transformações geométricas, estas operações requerem aritmética simples, tais como, adições de inteiros, deslocamentos de bits e comparações [39]. A Figura 2.4a ilustra a idéia da representação *octree* de uma cena. A cena é dividida em oito subcenas iguais, que são identificados por números associados (Figura 2.4b). O processo de subdivisão termina para as subcenas que contêm apenas voxels não pertencentes ao objeto (nós vazios) ou voxels pertencentes ao objeto (nós cheios). No caso de subcenas parciais, cada subcena é subdividida em oito subcenas novamente seguindo a mesma lógica de numeração como mostra a Figura 2.4c. O resultado final é a estrutura de árvore octal, onde os subcenas cheias e vazias representam as folhas e os subespaços parciais representam os nós com oito ramos cada.

Duas outras representações digitais são a estrutura shell [6] e o codificação da cena em corridas transparentes e não-transparentes [9, 13]. Estas abordagens serão deixadas para o próximo capítulo, onde estas estruturas serão apresentadas no contexto dos métodos *shell rendering* e *shear-warp rendering*.

2.5 *Rendering*

O *rendering* simula a propagação de luz em um ambiente de visualização para formar diferentes vistas dos objetos da cena. No caso de *rendering* de superfície, onde os voxels são opacos, a luz incidente é sempre refletida

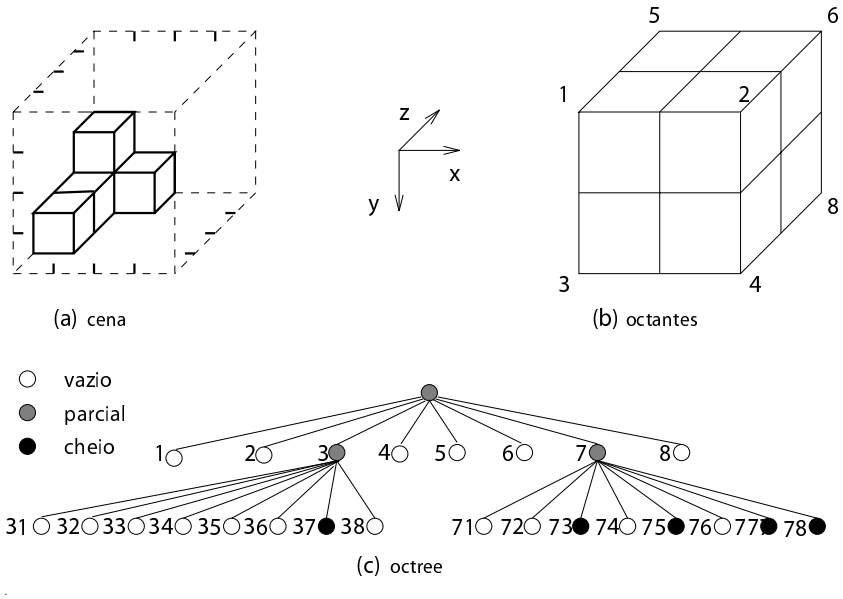


Figura 2.4: A cena em (a) é subdividida em octantes em (b), representados por uma *octree* em (c).

e nunca transmitida pelo voxel. No caso de *rendering* de volume, onde as opacidades estão no intervalo [0,1], o *rendering* simula o efeito da reflexão e transmissão da luz através dos voxels. Em ambos os casos a imagem resultante codifica no brilho dos pixels a parcela de luz refletida que chega aos olhos do observador. As transformações geométricas são usadas para simular os movimentos da cena com relação ao observador. Estas transformações são rotação e translação. O *rendering* propriamente dito consiste de projeção da cena no plano de visualização para uma dada posição relativa entre observador e cena, e uma tonalização dos voxels projetados.

2.5.1 Ambiente de visualização

O ambiente de visualização contém o observador, a cena e fontes de luz. O modelo de ambiente utilizado neste trabalho é apresentado na Figura 2.5. Neste ambiente o observador e uma fonte de luz branca estão em uma posição $(0, 0, -\infty)$, sobre o semi-eixo $z-$ e o plano de visualização é paralelo ao plano xy e posicionado em $(0, 0, d/2)$, onde d é a diagonal da cena. Assume-se que essa cena pode ser rotacionada em torno do seu centro na frente do observador, que permanece em uma posição fixa para qualquer direção de

visualização.

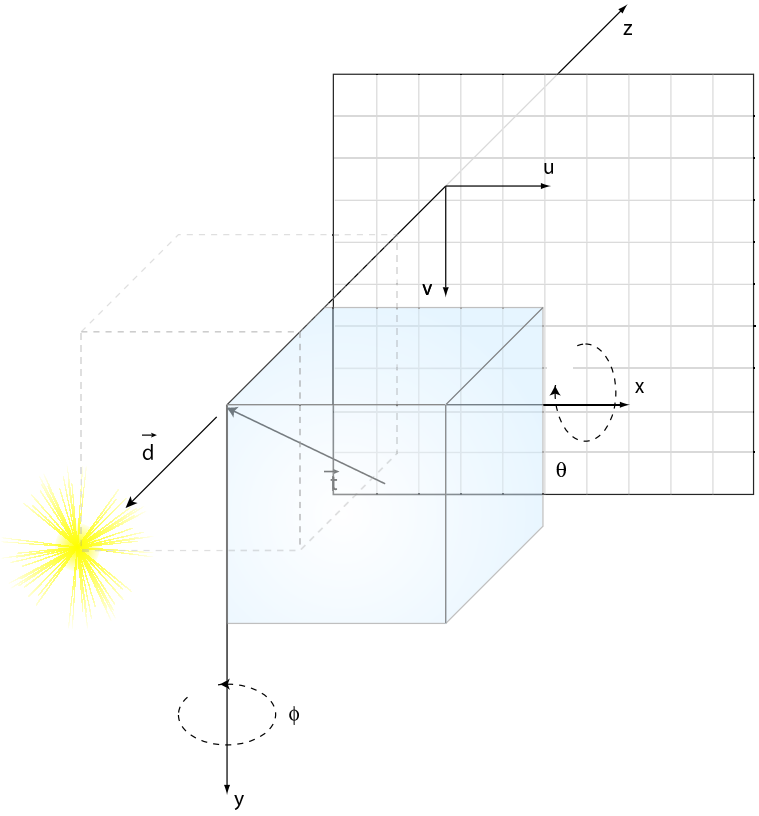


Figura 2.5: O ambiente de visualização utilizado neste trabalho é constituído pelo observador e uma única fonte de luz posicionados em $(0, 0, -\infty)$ e uma cena, centralizada na origem do sistema de coordenadas.

2.5.1.1 Translação

A rotação da cena nesse ambiente de visualização requer uma translação para que seu centro coincida com a origem do sistema de coordenadas. Esta operação de translação consiste de um deslocamento de todos voxels da cena em uma direção fixa $\vec{t}$. Para tanto, a cada coordenada x, y, z de voxel são somados os respectivos valores t_x, t_y e t_z através da matriz M_T. Quando a translação $\vec{t} = (-n/2, -m/2, -p/2)$ é aplicada na cena da Figura 2.1, esta fica centralizada na origem do sistema de coordenadas, como mostra a Figura 2.5. Após a formação da imagem no plano de visualização, uma

translação de $\vec{t} = (d/2, d/2, 0)$ na imagem torna as coordenadas dos pixels positivas.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = M_T \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.4}$$

$$M_T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

2.5.1.2 Rotação

Esta transformação é aplicada a cada coordenada (x, y, z) de cada voxel, definindo novo posicionamento (x'_i, y'_i, z'_i). Inicialmente os voxels têm o posicionamento absoluto definido em relação ao sistema de coordenadas, conforme o modelo apresentado na Figura 2.5. Neste modelo, a cena pode ser transladada e rotacionada. A translação centraliza a cena na origem do sistema de coordenadas e a rotação define dois ângulos θ e ϕ em torno dos eixos x e y, respectivamente. Desta forma, a rotação é sempre aplicada em torno do centro da cena e permite a visualização de qualquer face da cena através da variação dos ângulos θ e ϕ. A seguinte equação representa a transformação geométrica da rotação dos voxels da cena:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = M_R \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.6}$$

A matriz M_R é a composição das conhecidas matrizes M_θ e M_ϕ de rotação em torno dos eixos x e y, respectivamente (Equações 2.7 e 2.8). A ordem das matrizes na Equação 2.9 indica que a rotação é feita inicialmente em torno do eixo y e em seguida em torno do eixo x.

$$M_\theta = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\theta) & sen(\theta) & 0 \\ 0 & -sen(\theta) & cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

$$M_\phi = \begin{bmatrix} cos(\phi) & 0 & -sen(\phi) & 0 \\ 0 & 1 & 0 & 0 \\ sen(\phi) & 0 & cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

$$M_R = M_\theta \cdot M_\phi \tag{2.9}$$

Portanto,

$$M_R = \begin{bmatrix} cos(\phi) & 0 & -sen(\phi) & 0 \\ sen(\theta)sen(\phi) & cos(\theta) & sen(\theta)cos(\phi) & 0 \\ cos(\theta)sen(\phi) & -sen(\theta) & cos(\theta)cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

2.5.2 Projeção

A projeção dos voxels no plano de visualização é uma transformação geométrica aplicada em $\mathcal{R}^3$ gerando coordenadas em $\mathcal{R}^2$. Os modelos de projeção podem ser divididos em projeção perspectiva e ortogonal. A projeção em perspectiva é usada em aplicações de endoscopia virtual, onde o observador passeia no interior da estrutura em estudo. A maioria das aplicações usa a projeção ortogonal por ser mais simples, mais eficiente e atender os objetivos da visualização. O modelo de projeção utilizado neste trabalho é o modelo de projeção ortogonal.

Em projeção ortogonal as linhas formadas pelo percurso do raio de luz são paralelas entre si e ortogonais ao plano de visualização. Isto é simulado com o observador posicionado no infinito conforme o ambiente de visualização da Figura 2.5. Como resultado, objetos mais afastados do observador aparecem na imagem final com mesmo tamanho de objetos mais próximos. As coordenadas (x', y') em R^2 (Equação 2.11) são obtidas nas duas primeiras linhas da matriz de visualização M_{view} (Equação 2.12):

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = M_{view} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.11}$$

Onde:

$$M_{view} = M_R \cdot M_T = \begin{bmatrix} cos(\phi) & 0 & -sen(\phi) & -t_x \\ sen(\theta)sen(\phi) & cos(\theta) & sen(\theta)cos(\phi) & -t_y \\ cos(\theta)sen(\phi) & -sen(\theta) & cos(\theta)cos(\phi) & -t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.12}$$

Os tipos de projeção ortogonal podem ser divididos em duas classes principais: *ray casting* e *voxel splatting* [13]. As duas características que distinguem cada classe são: a ordem em que cada algoritmo atravessa a cena e o método que o algoritmo usa para projetar voxels na imagem. Cada classe de algoritmo tem vantagens e desvantagens em desempenho, e além disto, uma classe têm técnicas de aceleração mais simples que outra.

A taxonomia utilizada é baseada em laços de varredura. Grande parte dos algoritmos de *rendering* consistem de seis laços encadeados: três que fazem a iteração na cena e três que fazem a iteração no filtro de reamostragem.

2.5.2.1 *Ray casting*

Em *ray casting*, o ambiente de visualização é definido com o plano de visualização entre a cena e o observador, como mostra a Figura 2.6. Algoritmos *ray casting* produzem uma imagem através do lançamento de um raio através da cena para cada pixel e integrando o brilho e opacidade ao longo do raio [38, 40, 41]. Algoritmos *ray casting* são chamados *image order* pois a iteração dos laços mais externos é feita na imagem. Os laços mais internos utilizam uma função que extrai as coordenadas dos voxels e sua contribuição parcial na reamostragem linear nos eixos x,y e z.

Para $y_i \leftarrow 1$ **até** Imagem.altura
  Para $x_i \leftarrow 1$ **até** Imagem.comprimento
    Para $z_i \leftarrow 1$ **até** Imagem.profundidade
      Para cada x na Reamostragem(x_i, y_i, z_i)
        Para cada y na Reamostragem(x_i, y_i, z_i)
          Para cada z na Reamostragem(x_i, y_i, z_i)
            Adicione a contribuição do Voxel(x, y, z) no Pixel(x_i, y_i)

Os dois laços mais externos fazem a iteração nos pixels da imagem. O próximo laço faz a amostragem de pontos ao longo do raio no espaço imagem.

Finalmente, os três laços internos fazem a iteração sobre os voxels necessários na reamostragem para reconstruir uma amostra no espaço imagem. O corpo do laço multiplica o valor de cada voxel por um peso de reamostragem e adiciona o resultado no pixel correspondente na imagem.

Algoritmos *ray casting* são também chamados de algoritmos de projeção inversa, pois eles calculam a mapeamento de voxels para pixels da imagem, partindo dos pixels.

A maior desvantagem deste tipo de algoritmo é que ele não acessa a cena na ordem de armazenamento, uma vez que os raios de visualização atravessam a cena em uma direção arbitrária. Como resultado, este algoritmo desperdiça mais tempo calculando a posição dos pontos de amostragem e fazendo o endereçamento aritmético do que a outra classe de algoritmos.

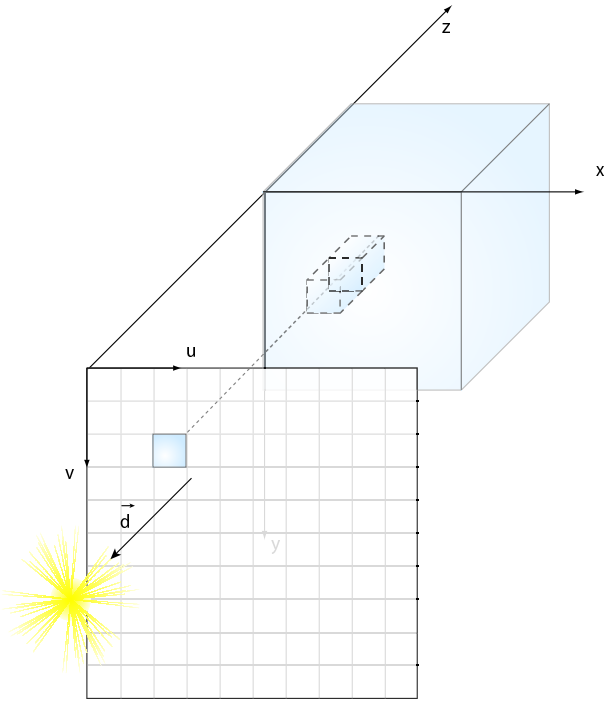


Figura 2.6: Em *ray casting*, o plano fica entre a cena e o observador. Neste algoritmo, uma imagem é formada pelo lançamento de um raio através cena para cada pixel.

2.5.2.1.1 Remoção de superfícies escondidas Um aspecto importante na projeção é a técnica de remoção de superfícies escondidas. Es-

ta técnica visa tonalizar apenas os voxels visíveis pelo observador, fazendo composição da tonalização dos voxels que pertencem a um mesmo raio e ao mesmo tempo calculando a opacidade acumulada no raio, conforme será descrito na seção 2.5.3.

Em *ray casting*, as superfícies escondidas são removidas durante o cálculo da opacidade acumulada do raio que atravessa a cena. Desta forma, o algortimo utiliza força bruta, ou seja, não possui nenhuma otimização do caminho pecorrido pelo raio. O objetivo dessa otimização é reduzir o cálculo em regiões ocultas da cena. Entre as técnicas utilizadas destacam-se *heirarchical enumeration* e *early ray termination*.

Heirarchical enumeration utiliza uma estrutura de dados *octree* que subdivide sucessivamente cena, formando uma árvore com diversos níveis. Um nível é percorrido apenas se este possui voxels não-transparentes. Caso contrário, cada nível da árvore que possui voxels transparentes pode ser omitido do processo de busca do raio. Em *early ray termination*, a otimização do *rendering* é terminar os cálculos de tonalização de um pixel quando a opacidade acumulada alcança um valor limiar que corresponde a uma parcela ínfima de luz refletida para observador. Levoy reporta que um algoritmo de *ray casting* em cena de dados médicos, com limiar de 95%, consegue acelerações de 1,6 a 2,2 vezes. Quando combinado com uma *octree*, esta aceleração é de 5 a 11 vezes [7].

2.5.2.2 *Voxel Splatting*

Ao contrário dos algoritmos de *ray casting*, algoritmos de *voxel splatting* operam através da iteração nos voxels da cena [42]. Esta classe de algoritmos calcula a contribuição de cada voxel para a imagem através de sua projeção em uma vizinhança de pixels que distribui o valor de tonalização do voxel de acordo com o filtro utilizado. Esta vizinhança envolve mais de um pixel quando o voxel ou a máscara do filtro possui tamanho maior que um pixel. Algoritmos deste tipo são chamados de *object order*, pois o laço mais externo faz a iteração com os voxels do objeto que está sendo construído na imagem.

Para $z \leftarrow 1$ **até** Cena.profundidade
  Para $y \leftarrow 1$ **até** Cena.altura
    Para $x \leftarrow 1$ **até** Cena.comprimento
      Para cada y_i na Projeção(x, y, z)
        Para cada x_i na Projeção(x, y, z)
          Adicione contribuição do Voxel(x, y, z) no Pixel(x_i, y_i)

Os três laços externos fazem a iteração sobre os voxels. Finalmente, os laços mais internos fazem a iteração sobre os pixels afetados por um voxel.

Este tipo de algoritmo é também chamado de algoritmo de projeção direta, pois os voxels são projetados diretamente na imagem, na mesma direção dos raios de visualização. Uma vantagem sobre o algoritmo *ray casting* é que a varredura em *object order* é feita na ordem de armazenamento. A vizinhança deve ser escolhida de modo a evitar "buracos" (Figura 2.11) ou sobreposição excessiva entre voxels adjacentes após a projeção na imagem.

2.5.2.2.1 Remoção de superfícies escondidas Durante a projeção da cena é necessário evitar a projeção de voxels que não são visíveis pelo observador. A remoção de superfícies escondidas consiste em identificar sobre cada raio de projeção, o ponto a partir do qual as contribuições dos voxels que pertencem ao objeto, projetados sobre um pixel p_i, deixam de influenciar na composição do brilho desse pixel. Isto ocorre quando a opacidade acumulada do pixel p_i está saturada, isto é, quando a luz refletida por um voxel v_i não consegue alcançar o pixel sobre o qual é projetada porque a soma das opacidades dos voxels entre v_i e o observador é maior ou igual a 1 (assumindo o ambiente da Figura 2.5). Dentre as técnicas de remoção de superfícies escondidas usadas com *voxel splatting* de voxels [43], as mais comuns são: *depth-sort*, *z-buffer*, *back-to-front* e *front-to-back*:

- **z-buffer**: algoritmo bastante difundido em computação gráfica. Para cada pixel do plano de visualização são associados dois valores: um valor de tonalização e o valor da distância (medida sobre o raio de visualização) entre o voxel que está sendo projetado e o pixel. Um novo valor de tonalização e um novo valor de distância são associados ao pixel apenas se o novo valor de distância for maior que o anterior. Isto garante que, entre os voxels não-nulos interceptados pelo raio de visualização, o voxel visível é o que está mais próximo do observador. Fica claro também que este método, da forma como está descrito acima, pode ser utilizado apenas em *rendering* de superfícies, uma vez que a tonalização do pixel não é calculada pela combinação da contribuição de vários voxels, mas é considerada apenas a contribuição do voxel mais próximo ao observador.

- **depth-sort**: trata-se de um outro algoritmo bastante comum em computação gráfica. Ele classifica todos os voxels da cena de acordo com

a distância ao plano de visualização, e depois projeta-os na ordem em que a distância decresce; do mais afastado para o mais próximo do plano. Esse processo de classificação torna-o mais lento e ligeiramente mais complexo que o algoritmo *z-buffer*. Porém, este método permite que o brilho de um pixel seja calculado pela combinação das contribuições de vários voxels. Logo, é adequado também para *rendering* de volume.

- **back-to-front**: o algoritmo *back-to-front* (BTF) [38, 39, 44] explora a informação de proximidade com o observador, implícita na cena (coluna-por-coluna, linha-por-linha, fatia-por-fatia), evitando a utilização de qualquer tipo de classificação de distâncias. Este algoritmo acessa os voxels coluna-por-coluna, linha-por-linha, fatia-por-fatia, na ordem em que eles se aproximam do observador, fazendo com que os mais próximos sejam projetados, no plano de visualização, sobre os mais afastados. Uma desvantagem deste algorítmo é que ele somente remove superfícies de cenas totalmente opacas, ou seja, não pode ser utilizado para *rendering* de volume.

- **front-to-back**: o algoritmo *front-to-back* (FTB) baseia-se na mesma idéia do algoritmo anterior, mas acessa os voxels na ordem em que eles se afastam do observador. A vantagem do acesso FTB [43, 44] em relação ao acesso BTF é que ele pode acelerar o processo através de técnicas de otimização, como evitar o cálculo de tonalização para pixels da imagem final que já tem suas opacidades saturadas. Os métodos descritos nos próximos capítulos usam a técnica FTB. Portanto, pata tornar mais clara sua definição apresenta-se um exemplo 2D na Figura 2.7.

A implementação eficiente de *early ray termination* é trivial para um algoritmo de *ray casting*, mas o mesmo não é verdadeiro em *voxel splatting*. Uma aproximação de *early ray termination* é checar a opacidade de cada pixel na imagem antes de fazer a sua composição com o voxel. No entanto, o algoritmo ainda teria que visitar cada voxel da cena. Esta otimização reduz o número de operações de tonalização, reamostragem e composição, mas não reduz o tempo gasto com regiões ocultas da cena com transformações geométricas e projeção. Estas operações em sua maioria envolvem multiplicações ponto-flutuante que têm grande efeito no tempo de processamento do *rendering*, sendo necessário evitá-las quando possível.

Estruturas de dados baseadas em *octree* e *quadtree* na imagem podem reduzir significativamente a complexidade do algoritmo da cena [45]. Ao

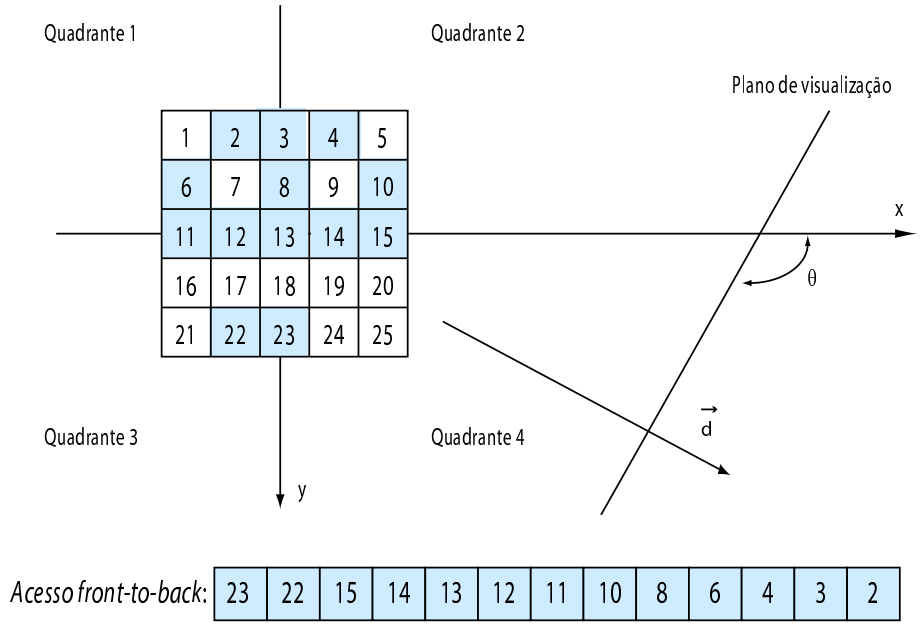


Figura 2.7: Um exemplo de uma cena 2D 5×5 com voxels numerados de 1 a 25. Os voxels opacos estão sombreados. A varredura é feita na ordem dos eixos: x e y, do voxel mais próximo do observador para o mais afastado.

invés de examinar cada voxel, o algoritmo pode evitar um nó oculto na *octree* da cena checando um pequeno número de nós na *quadtree* da imagem que já tem opacidade saturada.

Reynolds et al. [46] propõem uma técnica mais eficiente, chamada de *dynamic screen*, para implementação de *early ray termination* em um algoritmo de *rendering* de volume com projeção *object-order*. Ao invés de usar uma *octree* e uma *quadtree*, esta técnica utiliza um código de corrida para codificar a coerência espacial da cena e da imagem. Os autores fazem a seguinte observação: se uma transformação é uma projeção paralela então ela transforma linhas paralelas no espaço da cena em linhas paralelas no espaço da imagem. Esta observação leva a um algoritmo de *rendering* eficiente: para cada linha de varredura de voxels (em projeção *front-to-back*) o algoritmo atravessa simultaneamente a linha de varredura na cena e a linha projetada na imagem.

2.5.3 Tonalização e Cor

A tonalização tem por objetivo fornecer a impressão 3D associando a cada pixel da imagem um brilho e, alternativamente, uma cor utilizada para identificar objetos distintos na imagem ou criar uma impressão de realismo [47, 43].

A tonalização, associada a um pixel p_i da imagem, é resultado da combinação da luz e da opacidade provenientes de todos os voxels $v_1, v_2, ..., v_n$, que são projetados em p_i. A combinação dos voxels deve levar em conta a opacidade acumulada a cada passo do processo, pois se essa opacidade acumulada atinge um valor muito próximo do valor de opacidade máxima, pode-se considerar que a luz refletida por qualquer outro voxel mais afastado não chegará ao pixel, pois irá perdendo intensidade, conforme atravessa os voxels mais próximos do observador, até que nada reste para contribuir na tonalização do pixel. Nessa situação, o pixel p_i pode ser considerado saturado.

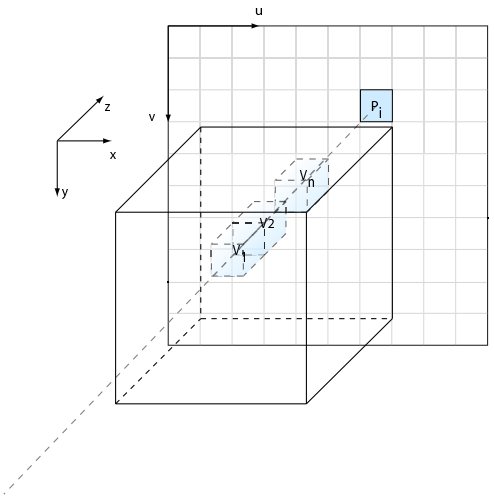


Figura 2.8: A tonalização é resultado da combinação da luz e da opacidade provenientes de todos os voxels $v_1, v_2, ..., v_n$, que são projetados em p_i.

A equação abaixo mostra uma forma como a tonalização associada ao pixel p_i pode ser estimada:

$$S_{p_i} = \sum_{i=1}^{n} I_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \tag{2.13}$$

O componente $I_i\alpha_i$, da equação 2.13, representa o total de luz que atravessa um voxel v_i, onde I_i é o valor da luz incidente neste voxel, calculado através de algum modelo de iluminação conhecido, como por exemplo o modelo de Gouraud [48] ou Phong [49]; e α_i é o valor da opacidade de v_i (esse valor está no intervalo [0, 1], de forma que se $\alpha_i = 1$, toda a luz incidente é refletida).

A luz refletida por um voxel v_i, antes de atingir o pixel sobre o qual está sendo projetada, deve atravessar o meio semi-transparente de todos os outros voxels que estão na mesma direção de projeção. Esse obstáculo faz com que a luz total, refletida por v_i, perca intensidade conforme viaja através desses voxels. A equação 2.13 expressa esse efeito através do componente $\prod_{j=1}^{i-1}(1-\alpha_j)$, que representa a parcela de luz refletida por v_i que atravessa os voxels v_j, $j = 1, 2, \ldots, i-1$, que estão entre v_i e o observador.

O modelo de iluminação mais difundido na área de visualização de imagens médicas [47] é o modelo de iluminação de Phong. Em Phong, algumas propriedades do comportamento da luz são levadas em conta, para definir como a luz ambiente e a fonte de luz influenciam no cálculo do valor de tonalização:

- **Influência da distância da fonte de luz:** A intensidade de luz, refletida por cada voxel em direção ao observador, cai com o aumento da distância entre esse voxel e o observador. Este efeito é representado pela variação do valor de tonalização, em função da distância normalizada entre o voxel e o plano de visualização (valor do *z-buffer*). Quanto mais afastado do plano de visualização, mais claro será esse ponto, e quanto mais próximo do plano, mais escura será sua tonalização. Esse efeito é modelado de forma linear pela equação 2.14.

$$I_{dist}(u,v) = \frac{I_{max} - I_{min}}{d_{max} - d_{min}}\,[d - d_{min}] + I_{min} \qquad (2.14)$$

  onde:
  I_{max} é a intensidade máxima desejada na imagem;
  I_{min} é a intensidade mínima desejada na imagem;
  d_{max} é a maior distância possível entre um voxel e o plano de visualização (maior valor que z' pode assumir);
  d_{min} é a menor distância possível entre um voxel e o plano de visualização (menor valor que z' pode assumir);
  d é a distância do voxel v até o observador.

- **Influência da Fonte de Luz:** a luz proveniente de uma fonte pontual, incidindo sobre um voxel, resulta em duas formas de reflexão que são consideradas no modelo de iluminação de Phong:

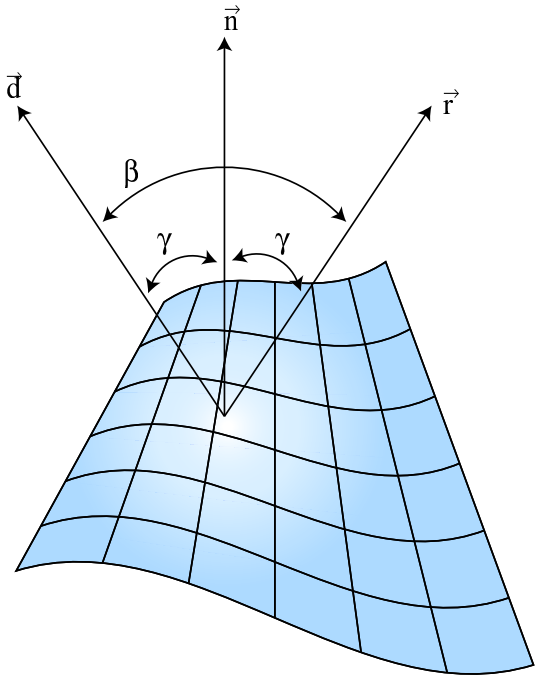


Figura 2.9: Modelo de Phong de iluminação.

  - **Reflexão Difusa:** Parcela da intensidade de luz refletida em um voxel é a mesma em todas as direções, portanto atingindo o observador independente de sua posição, mas varia com o cosseno do ângulo γ entre a direção do raio de luz incidente e a direção do vetor normal à superfície, estimado neste voxel.

  - **Reflexão Especular:** Parcela da intensidade de luz, refletida em um voxel, que é máxima na direção $\vec{r}$ de reflexão, que forma um ângulo γ com o vetor normal à superfície, no lado oposto ao da luz incidente, e vai diminuindo com o afastamento em relação à direção de visualização $\vec{d}$, de acordo com alguma regra (Figu-

ra 2.9). Esta intensidade varia com o cosseno do ângulo β, entre a direção $\vec{d}$ e a direção do observador, elevado a um expoente n, relativo ao tipo de tecido. Como as direções do observador e da fonte de luz são iguais, $\beta = 2\gamma$.

- **Influência à luz ambiente:** Parcela da intensidade de luz, refletida pelos voxels, que é constante devido à luminosidade natural do ambiente, e é modelada por um fator constante K_a.

Uma vez definidas tais propriedades, pode-se representar o modelo de iluminação de Phong através da seguinte equação:

$$I(u,v) = I_{max}K_a + I_{dist}(u,v)(K_d cos(\gamma) + K_s cos^n(\beta)) \tag{2.15}$$

onde:
I_{max} é a intensidade máxima desejada na imagem;
$I(u,v)$ é o nível de cinza associado ao pixel (u, v);
K_a é o coeficiente de reflexão para a luz ambiente;
K_d é o coeficiente de reflexão difusa;
K_s é o coeficiente de reflexão especular;
n é um coeficiente do material;
$I_{dist}(u,v)$ é o valor da tonalização baseada em distância, calculado através da (equação 2.14);
γ é o ângulo entre o vetor normal à superfície e o raio de incidência. Devem ser considerados os valores de γ entre 0 e 90 graus para o componente difusa, e valores entre 0 e 45 graus para o componente especular.
β é o ângulo entre o raio de incidência e a direção de reflexão D, e como a posição do observador e da fonte de luz são iguais, o valor de β é igual a 2γ.

Os valores dos parâmetros da equação 2.15 são escolhidos pelo usuário de forma empírica, até a obtenção de um resultado satisfatório. Os coeficientes de reflexão (K_d, K_s, K_a) devem apresentar valores no intervalo [0, 1], de forma que a soma deles seja igual a 1. O expoente n, relativo ao tipo de tecido, deve ter um valor no mínimo igual a 0. Os cossenos dos ângulos β e γ são obtidos pelo produto interno entre os vetores normalizados nas respectivas direções.

2.5.3.1 Estimação das normais à superfície dos tecidos

As técnicas de visualização, em sua maioria, requerem também que um vetor normal seja associado a cada voxel. O vetor normal é utilizado para o cálculo de tonalização durante a projeção.

A direção da normal em cada voxel pode ser estimada de diversas formas: do gradiente na cena original, o *z-buffer*, gradiente da cena de opacidades, ou o gradiente da cena binária filtrada do mapa de distância Euclidiana. Devido à sua influência na qualidade da imagem, deve-se escolher com muito cuidado o método utilizado para estimar as normais. A Figura 2.10 ilustra uma vizinhança do tipo 6 de voxels, para o cálculo dos vetores normais à superfície do objeto a partir do gradiente da cena. Esse cálculo pode ser feito da seguinte forma:

$$g_x = \frac{f(x+1,y,z) - f(x-1,y,z)}{2}$$

$$g_y = \frac{f(x,y+1,z) - f(x,y-1,z)}{2}$$

$$g_z = \frac{f(x,y,z+1) - f(x,y,z-1)}{2}$$

$$\vec{n} = \frac{\vec{g}}{||\vec{g}||} \tag{2.16}$$

onde:

$\vec{g}$ é o vetor gradiente da cena de entrada.
g_x, g_y, g_z são os componentes do vetor gradiente nos eixos x, y e z.
$f(x,y,z)$ é o nível de cinza na cena de entrada;
$\vec{n}$ é o vetor normal estimado na superfície do objeto, normalizado.

A normal proveniente do cálculo do gradiente da cena depende da constituição da estrutura que se deseja visualizar. Por exemplo, o gradiente na cena original é recomendado para tecidos ósseos em CT, mas não para estes mesmos tecidos em MRI. Como depende muito da modalidade de aquisição, esta solução não é indicada para todos os casos.

Em cenas binárias, a normal estimada não reflete com fidelidade a normal da superfície do objeto pois a cena apresenta variações abruptas de intensidade em uma vizinhança local. A Figura 2.10 mostra um exemplo de uma vizinhança-6, onde a normal é estimada através do gradiente (Equação 2.16)

da intensidade dos voxels. A suavização das bordas do objeto através de uma filtragem Gaussiana da cena binária diminui a diferença entre a normal calculada e normal da superfície do objeto, mas não é uma solução eficaz para todos os casos.

O mapa de distância Euclidiana 3D na cena binária que contém o objeto de interesse torna o cálculo da normal independente da modalidade de aquisição. Esta alternativa é extremamente útil em tecidos em que o gradiente não é bem definido e/ou não existe uma borda suave. É a técnica que apresenta melhores resultados para cenas binárias.

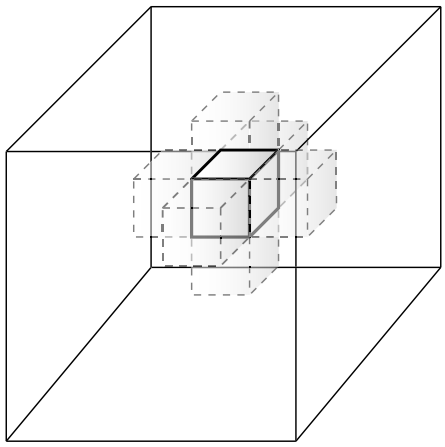


Figura 2.10: Vizinhança-6 de um voxel na cena.

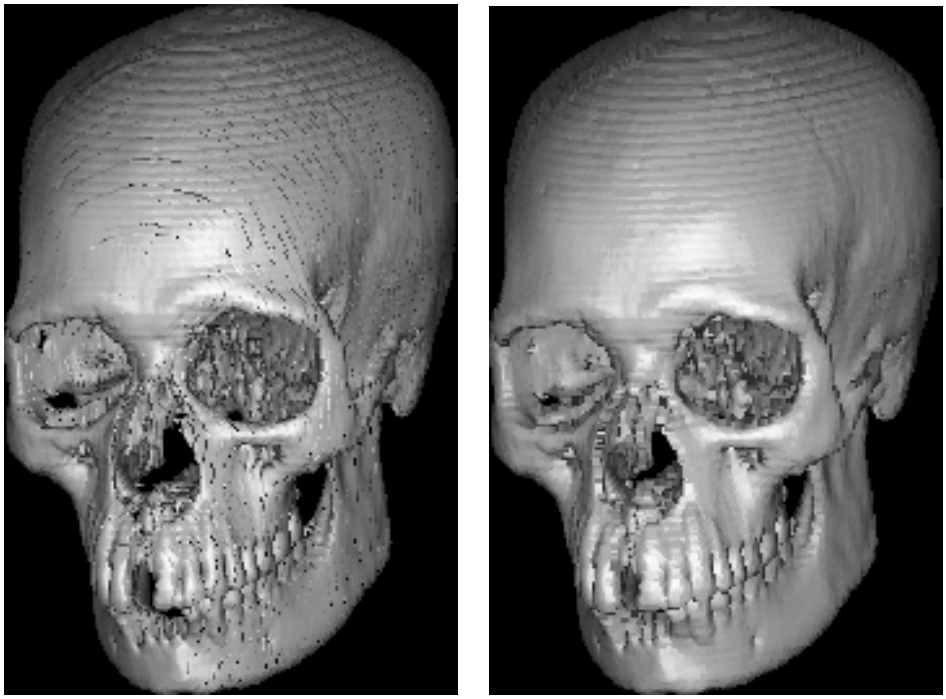
2.5.3.2 Quantização de normais

O armazenamento em ponto-flutuante do vetor normal na lista de atributos do voxel para visualização do voxel é bastante dispendioso em memória. Uma técnica de quantização é normalmente utilizada para diminuir o custo de memória. Desta forma, para cada voxel é associado o índice de vetor em uma tabela de normais. Esta tabela é construída através da amostragem de um grande número de normais na superfície de um sólido como, por exemplo, uma esfera ou um cubo. O voxel guarda o índice do elemento desta tabela que contém a normal que mais se aproxima à normal estimada em sua superfície. A quantização pode afetar a qualidade da imagem final quando o número de normais amostradas não é suficiente para gerar tons de cinza diferentes necessários para representar de forma satisfatória a variação de tonalidade na superfície dos tecidos.

2.5.4 Qualidade de imagem

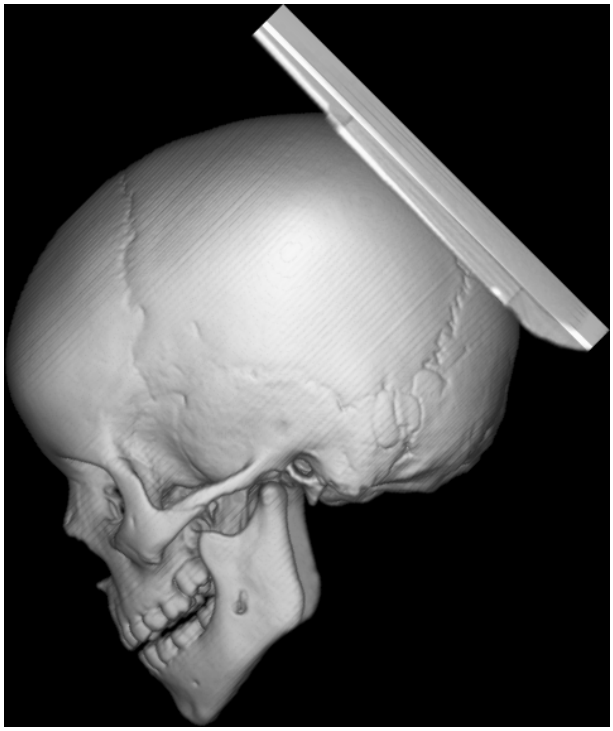
Métodos de visualização baseados em *voxel splatting* têm um problema bastante conhecido: o aparecimento de buracos na imagem gerada para algumas direções de visualização (veja Figuras 2.11a e 2.11b). Algoritmos de pós-processamento podem ser utilizados para "fechar" estes buracos, mais o sucesso desse algoritmos não é garantido e a imagem resultante pode apresentar baixa qualidade. Um solução mais eficiente é tornar o tamanho dos voxels maior que o tamanho de pixels durante a projeção *voxel splatting*.

Uma alternativa é utilizar *shear-warp rendering*, um algoritmo proposto na tese de doutorado de Philippe Lacroute [13] que combina vantagens de métodos *voxel splatting* e *ray casting* através da fatoração *shear-warp* da matriz de visualização. Além da fatoração são utilizadas em conjunto várias estratégias para otimizar o *rendering*, como a coerência espacial de estruturas da cena, *early ray termination* e a relação entre pixels do plano de visualização e voxels do volume.

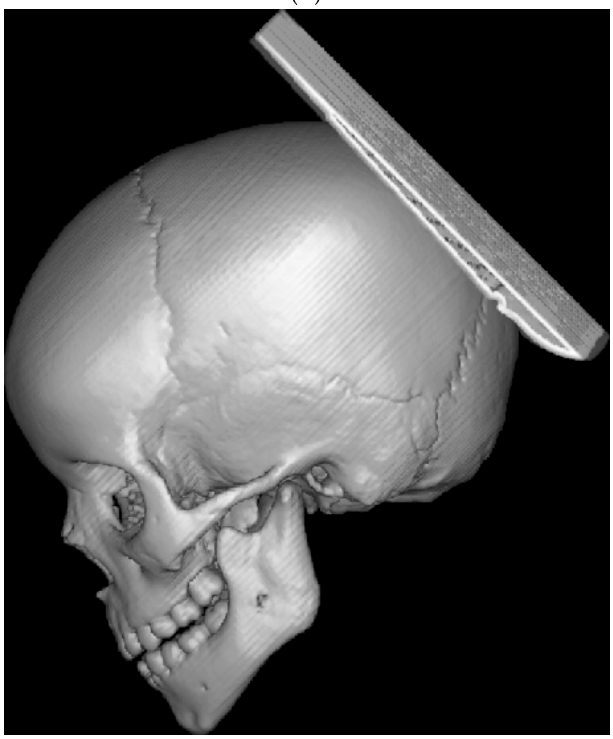


(a) (b)

Figura 2.11: *Rendering* de superfície usando *splatting* (a) de um voxel para um pixel e (b) de um voxel para 3×3 pixels.



(a)



(b)

Figura 2.12: Comparação entre imagens geradas pelo *rendering* com (a) fatoração *shear-warp* (b) *voxel splatting*

Capítulo 3

Conclusão

A despeito das dificuldades inerentes ao processo de classificação, visualização em velocidade interativa é sempre um desafio porque a quantidade de informação para visualização normalmente requer bastante espaço de memória e cálculos computacionais intensivos. A implementação do *rendering* de cenas pode feita através de bibliotecas gráficas como OpenGL que, a partir da versão 1.2, permite a criação de texturas 3D através da função *glTexImage3D*:

```
void glTexImage3D( GLenum target,
                   GLint level,
                   GLenum internalformat,
                   GLsizei width,
                   GLsizei height,
                   GLsizei depth,
                   GLint border,
                   GLenum format,
                   GLenum type,
                   const GLvoid *pixels )
```

Pesquisadores têm proposto *hardware* especializado [9, 10] e algoritmos que comprometem qualidade de imagem por velocidade [11, 12] na intenção de resolver o problema de velocidade. Os recentes processadores gráficos nVidia, a partir da série GeForce FX, permitem aceleração por *hardware* das texturas volumétricas. Uma alternativa é utilizar placas especializadas da família VolumePro, que permite o armazenamento de até 4Gb em memória, possibilitando *rendering* em tempo real de cenas de 512^3 voxels.

Enquanto o desenvolvimento de *hardware* dedicado é importante para avançar o conhecimento em computação, abordar o problema da interatividade através de algoritmos eficientes pode ser mais prático do ponto de vista de disponibilidade e portabilidade das implementações. Ainda mais, considerando que mais de 85% dos voxels são normalmente classificados como transparentes, as abordagens mais interessantes para o problema de velocidade usam estruturas de dados espaciais que codificam a presença e/ou ausência de voxels de alta opacidade para eliminar o cálculo em regiões transparentes da cena como também em regiões de alta opacidade [6, 7, 8, 13, 14]. Algumas das técnicas utilizadas com sucesso têm sido *shell rendering* [6] e *shear-warp rendering* [13].

Referências Bibliográficas

[1] J.K. Udupa. Three-dimensional imaging: Principles and approaches. Technical Report MIPG254, Medical Image Processing Group, Department of Radiology, University of Pennsylvania, Feb 1999.

[2] R. Gonzalez e R.R. Woods. *Digital Image Processing.* Addison-Wesley, New York, 1993.

[3] W.E. Lorensen e Cline H.E. Marching cubes: A high resolution 3D surface construction algorithm. *Computer & Graphics*, páginas 163–169, Jul 1987.

[4] M.J. Durst. Letters: Additional reference to marching cubes. *Computer Graphics (In Proc. of ACM SIGGRAPH)*, 22(2):72–73, Apr 1988.

[5] I. Gargantini e H. Atkinson. Multiple-seed 3D connectivity filling for innacurate borders. *CVGIP: Graphical Models and Image Processing*, 53(6):563–573, 1991.

[6] J.K. Udupa e D. Odhner. Shell rendering. *IEEE Computer Graphics and Applications*, 13(6):58–67, 1993.

[7] M. Levoy. Volume rendering by adaptive refinement. *The Visual Computer*, páginas 2–7, 1990.

[8] K.R. Subramanian e D.S. Fussell. Applying space subdivision techniques to volume rendering. Em *Proceedings of Visualization'90*, páginas 150–159, San Francisco, CA, 1990.

[9] P. Lacroute e M. Levoy. Fast volume rendering using a shear-warp factorization of viewing transformation. *Computer Graphics*, páginas 451–458, 1994.

[10] M. Bentum. *Interactive visualization of volume data.* PhD thesis, Dept. of Electrical Engeneering, University of Twente, Enschede, The Netherlands, Jun 1996.

[11] M. Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics*, páginas 245–261, 1990.

[12] J. Danskin e P. Hanrahan. Fast algorithms for volume ray tracing. Em *Proceedings of the 1992 Workshop on Volume Rendering*, volume 19, páginas 91–98, 1992.

[13] P. Lacroute. *Fast volume rendering using a shear-warp factorization of the viewing transformation.* PhD thesis, Dept. of Electrical Engineering and Computer Science, Stanford University, Stanford, CA, Sep 1995.

[14] J.K. Zuiderveld, A.H.J. Koning, e M.A. Viergever. Acceleration of ray-casting using 3d distance transforms. Em *Proceedings of Visualization in Biomedical Computing*, páginas 324–335, Chapel Hill, North Caroline, Oct 1992.

[15] J.D. Foley, S.K. Feiner A. van Dam, e J.F. Hughes. *Computer Graphics: Principles and Practice.* Addison-Wesley, New York, second ed. edition, 1990.

[16] J. Serra. Morphological filtering: An overview. *Signal Processing*, páginas 3–11, 1994.

[17] S. Raya. Low-level segmentation of 3-d magnetic resonance brain images. *IEEE Transactions on Medical Imaging*, páginas 327–337, 1990.

[18] L. Gong e C. Kulikowski. Composition of image analysis processes through object-centered hierarchical planning. *IEEE Transactions on Medical Imaging*, 53(6):563–573, 1991.

[19] D. Collins e T. Peters. Model-based segmentation of individual brain structures from mri data. Em *SPIE on Medical Imaging*, páginas 71–83, 1992.

[20] M. Sonka, S. Tadikonda, e S. Collins. Knowledge-based interpretation of mr brain images. Em *IEEE Transactions on Medical Imaging*, páginas 443–452, 1996.

[21] R. Bajecsy e S. Kovacic. Multiresolution elastic matching. *Computer Vision, Graphics and Image Processing*, páginas 1–21, 1989.

[22] J. Gee, C. Barillot, L. Le Briquer, D. Haynor, e R. Bajecsy. Matching structural images of the human brain using statistical and geometrical image features. Em *SPIE on Medical Imaging*, páginas 191–204, 1994.

[23] J. Talairach, G. Szikla, P. Tournoux, A. Prossalentis, M. Bordas-Ferre, L. Covello, M. Jacob, A. Mempel, P. Buser, e J. Bancaud. *Co-Planar Stereotaxis Atlas of the Human Brain: 3-Dimensional Proportional System - An Approach to Cerebral Imaging*. New York: Thieme Med., New York, 1998.

[24] J. Gee, M. Reivich, e R. Bajesy. Elastically deforming 3d atlas to match anatomical brain. *Journal of Computer Assited Tomography*, páginas 225–236, 1993.

[25] G. Christensen, R. Rabbitt, e M. Miller. 3-d brain mapping using a deformable neuroanatomy. *Physics in Medicine Biology*, 39:609–518, 1994.

[26] M. Kamber, R. Shinghal, D. Collins, G. Francis, e A. Evans. Model-based 3d segmentation of multiple sclerosis lesions in magnetic resonance brain images. Em *IEEE Transactions on Medical Imaging*, volume 14, páginas 442–453, 1995.

[27] D. Collins e T. Peters. A dual probabilistic classifier for three-dimensional neuroimaging from mri data. Em *SPIE on Medical Imaging*, páginas 373–384, 1994.

[28] E. Artzy, G. Frieder, e G. Herman. The theory, design, implementation and evaluation of a three-dimensional surface detection algorithm. *Computer Graphics and Image Processing*, 15:1–24, 1981.

[29] J. Udupa, S. Srihari, e G. Herman. Boundary detection in multidimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4:41–50, 1982.

[30] J. Canny. A computacional approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.

[31] H. Liu. Two- and three-dimensional boundary detection. *Computer Graphics and Image Processing*, 8:123–134, 1977.

[32] S.P. Raya e J.K. Udupa. Shape-based interpolation of multidimensional objects. *IEEE Transactions on Medical Imaging*, páginas 32–42, 1990.

[33] G.T. Herman, J. Zheng, e C.A. Bucholtz. Shape-based interpolation. *IEEE Computer Graphics and Applications*, páginas 69–79, 1992.

[34] W.E. Higgins, C. Morice, e E.L. Ritman. Shape-based interpolation technique for three-dimensional images. *Proceedings of IEEE 1990 International Conference on Acoustics, Speech and Signal Processing*, páginas 1841–1844, April 1990.

[35] R.A. Lotufo, G.T. Herman, e J.K. Udupa. Combining shape-based interpolation and gray-level interpolations. Em *SPIE Proceedings in Visualization in Biomedical Computing*, volume 1808, páginas 289–298, 1992.

[36] R.A. Drebin, L. Carpenter, e P. Hanrahan. Volume rendering. *Computer Graphics*, páginas 65–74, Aug 1988.

[37] A.R. Smith. Volume graphics and volume visualization: A tutorial. Technical Memo 176, Pixar Inc., California, 1987.

[38] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, páginas 29–37, May 1988.

[39] G. Frieder, D. Gordon, e R. A. Reynolds. Back-to-front display of voxels based objects. *IEEE Computer Graphics and Applications*, páginas 52–60, Jan 1985.

[40] P. Sabella. A rendering algorithm for visualizing 3D scalar fields. *Computer Graphics*, páginas 51–58, Aug 1988.

[41] C. Upson e M. Keeler. V-BUFFER: Visible volume rendering. *Computer Graphics*, 22:59–64, 1988.

[42] L. Westover. Footprint evaluation for volume rendering. *Computer Graphics*, 24:367–376, 1990.

[43] J.K. Udupa. 3D visualization of images. Technical Report MIPG196, Medical Image Processing Group, Department of Radiology, University of Pennsylvania, Jun 1993.

[44] R. A. Reynolds. *Fast methods for 3D display of medical objects.* PhD thesis, Dept. of Computer and Information Sciences, University of Pennsylvania, May 1985.

[45] D.J. Meagher. Efficient synthetic image generation of arbitrary 3-d objects. Em *Proceedings of the IEEE Conference on Pattern Recognition and Image Processing*, páginas 473–478, 1982.

[46] R. Reynolds, D. Gordon, e L. Chen. A dynamic screen technique for shaded graphics display of slice-represented objects. *Computer Vision, Graphics and Image Processing*, páginas 275–298, 1987.

[47] A.X. Falcao. Visualização de volumes aplicada à área médica. Tese de mestrado, FEEC-UNICAMP, Feb 1993.

[48] H. Gouraud. Continous shading of curved surfaces. *IEEE Transaction of Computers*, páginas 623–629, 1971.

[49] B. T. Phong. Illumination for computer generated pictures. *Communications of the ACM*, páginas 311–317, Jun 1975.