Introdução à Visualização Volumétrica

ANSELMO CARDOSO DE PAIVA
Pontifícia Universidade Católica - PUC-Rio - Tecgraf
Universidade Federal do Maranhão – UFMA
email: paiva@tecgraf.puc-rio.br

ROBERTO DE BEAUCLAIR SEIXAS
Laboratório Nacional de Computação Científica - LNCC
email:tron@lncc.br

MARCELO GATTASS
Pontifícia Universidade Católica - PUC-Rio – Tecgraf
email:gattass@tecgraf.puc-rio.br
PUC-RioInf.MCC03/99 January, 1999

ABSTRACT: A volumetric data set is a collection of data in which each data has an associated location in the three-dimensional space. In this work we discuss the techniques that handle these data sets to generate arbitraries views of the volume. We show the main aplications that use these techniques, classify the algorithms, and discuss some implementation issues.

Keywords: computer graphics, volume rendering, ray casting, medical imaging.

Resumo: Um conjunto de dados volumétricos é uma coleção de dados amostrados em uma grade no espaço 3D. Neste trabalho apresentamos as técnicas que são usadas para gerar vistas arbitrárias destes conjuntos de dados. Mostramos as principais aplicações, classificamos os algoritmos e discutimos algumas características de implementação.

Palavras-chave: computação gráfica, *rendering* de volumes, *ray casting*, imagens médicas.

Sumário

Lista de Figuras

1. Introdução

Visualização é um termo relacionado aos métodos que permitem a extração de informações relevantes a partir de complexos conjuntos de dados, processo geralmente feito através da utilização de técnicas de computação gráfica e processamento de imagens.

> De acordo com (McCormick,1987), visualização é uma ferramenta para a interpretação de dados representados em computador e para a geração de imagens a partir de conjuntos de dados complexos e multidimensionais. Denomina-se visualização científica quando estes conjuntos de dados representam fenômenos complexos e o objetivo é a extração de informações científicas relevantes.

Uma das mais interessantes subáreas da visualização científica, que tem tido um rápido crescimento, é a visualização volumétrica. Visualização volumétrica é o conjunto de técnicas utilizadas na visualização de dados associados a regiões de um volume, tendo como principal objetivo a exibição do interior de objetos volumétricos, a fim de explorar sua estrutura e facilitar sua compreensão (McCormick,1987).

Estes dados, quando associados a regiões de volumes, são denominados dados volumétricos. Assim, podemos conceituar a visualização volumétrica como a classe de métodos de visualização relacionada com a representação, manipulação e visualização de conjuntos de dados volumétricos.

Figura 1.1 Exemplos de visualização volumétrica

Em termos gerais, a visualização volumétrica é o processo de *rendering*, realizado com o objetivo de obter uma melhor compreensão da estrutura contida nos dados volumétricos. Na maioria das vezes este conjunto de dados é definido em uma

grade tridimensional com um ou mais valores escalares, ou vetoriais, em cada ponto da grade.

Os dados volumétricos são geralmente tratados como uma matriz de elementos de volume, denominados *voxels* (análogo 3D de um *pixel). Voxels* são paralelepípedos, fortemente agrupados, formados pela divisão do espaço do objeto através de um conjunto de planos paralelos aos eixos principais desse espaço. Esses elementos não devem se interceptar, sendo de tamanho suficientemente pequeno se comparado às características representadas pelos dados volumétricos. Existe uma certa ambigüidade na literatura a respeito do que é o conjunto de pontos de um *voxel*. Em determinadas publicações, *voxel* é o hexaedro definido em torno do valor amostrado como ilustra a Figura 1.2(a). Em outras, o *voxel* é entendido como sendo o hexaedro cujos vértices são os valores amostrados (Figura 1.2(b)). Neste trabalho, partindo de uma analogia com o conceito de *pixel*, estamos referenciando como *voxel* uma tupla <i,j,k,S> que define um ponto amostrado do campo escalar na posição *(i,j,k)* e com valor *S* associado.

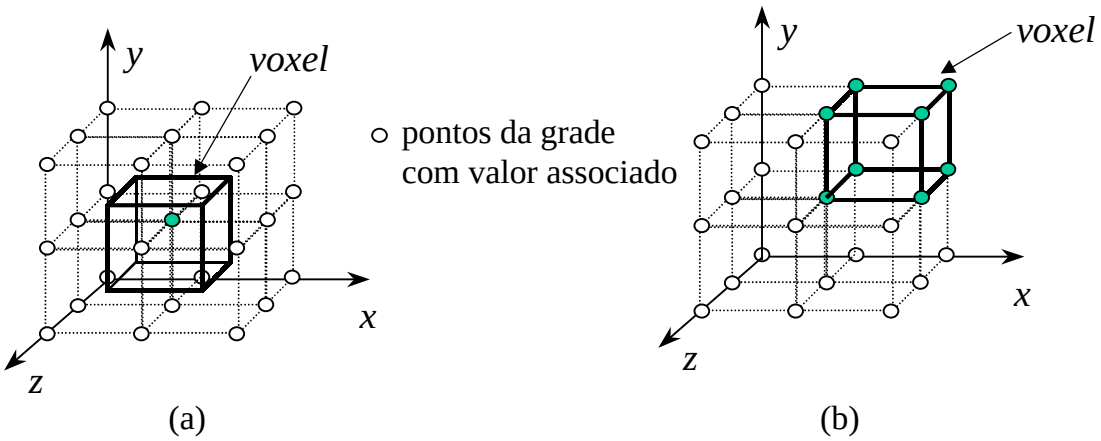


Figura 1.2 Elementos de volume

Em geral, podemos classificar os dados volumétricos como provenientes de duas fontes principais:

- Simulação – dados construídos a partir de algum modelo matemático;
- Aquisição – dados resultantes da conversão de objetos existentes na natureza.

As duas abordagens básicas para a solução do problema de visualização volumétrica são a extração de superfícies (*surface fitting*) e o *rendering* direto de volumes (*volume rendering*). Elas diferem basicamente pela utilização ou não de representações intermediárias dos dados volumétricos para a geração da visualização

adequada à aplicação. Enquanto no *rendering* direto de volumes a projeção é realizada diretamente a partir dos dados volumétricos, na extração de superfícies os dados volumétricos são convertidos para uma representação geométrica (polígonos) a partir da qual são usados os métodos tradicionais de *rendering* de polígonos para a geração da visualização. A Figura 1.3 apresenta uma idéia esquemática da conexão entre estas técnicas.

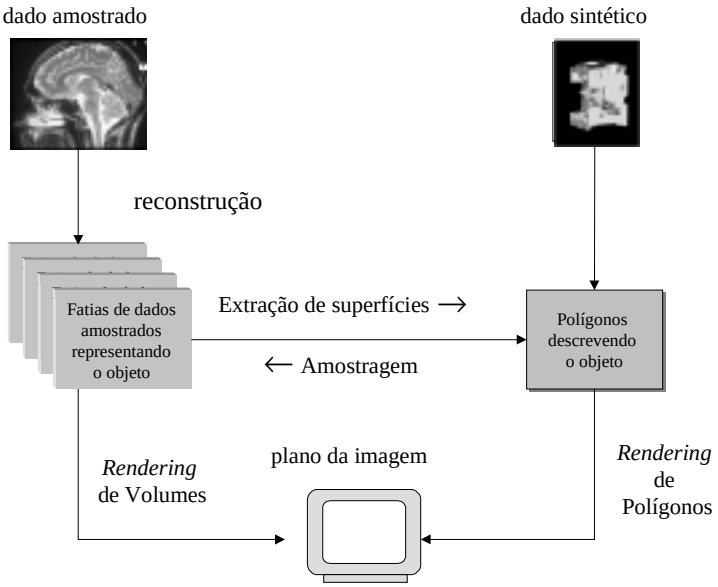


Figura 1.3 Representação do método de visualização volumétrica

As técnicas envolvidas no processo de visualização volumétrica (Figura 1.3) podem ser resumidas na execução de quatro passos básicos, definidos na Figura 1.4. No entanto, a visualização é realizada através da implementação apenas dos três últimos passos, pois consideramos apenas a visualização de volumes já pré-processados.

O passo de formação do volume envolve a aquisição propriamente dita dos dados, o pré-processamento das fatias e a reconstrução do volume. Os dados volumétricos podem ser oriundos de simulações baseadas em elementos finitos ou de modelos computacionais para dinâmica dos fluidos. Outra forma de aquisição desses dados é através da utilização de *scanners* sobre o material de interesse. Assim, podem ser utilizadas técnicas como Ressonância Magnética (MRI), Tomografia Computadorizada (CT), Tomografia por Emissão de Pósitrons (PET), Sismógrafos, etc. Além destas duas formas de aquisição, podemos acrescentar a geração de dados

volumétricos através da conversão de objetos geométricos para modelos de *voxels* (*voxelisation*).

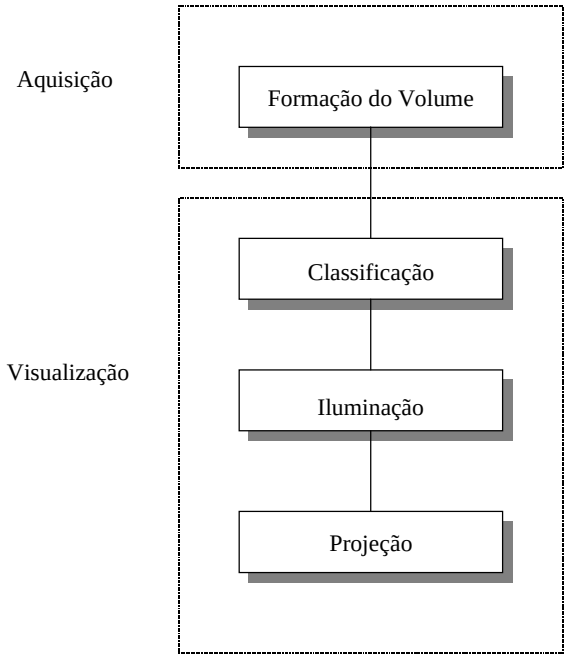


Figura 1.4 *Pipeline* de visualização volumétrica

Após a aquisição dos dados é necessário processar as fatias de dados de modo a melhorar características como contraste, relação sinal/ruído e faixa de variação dos valores.

No final do processo de formação do volume, os dados são reconstruídos, gerando um volume com dimensões proporcionais. Nesta fase, novas fatias podem ser geradas por interpolação ou duplicação para preencher vazios no volume e o conjunto de pontos amostrados pode ser convertido de um caso irregular (obtido na etapa de aquisição) para uma grade cartesiana regular.

Uma vez fornecido um volume já formado, podemos dar início ao processo de visualização. Para isto iniciamos com a etapa de classificação, que está relacionada à identificação do material representado em cada *voxel*. Esta etapa possibilita a seleção de características dos dados, segundo um critério quantitativo, definindo a região do volume que se deseja explorar. Em técnicas de Extração de Superfícies, classificar significa definir o valor de limiarização (*threshold*) utilizado para identificar a superfície que será poligonizada e posteriormente visualizada. Já para técnicas de *rendering* direto, a etapa de classificação envolve a definição da relação entre os

valores dos dados do volume e os valores de cor e opacidade que serão utilizados no algoritmo de exibição- ou seja, a definição das funções de transferência.

Em seguida é aplicado um modelo de iluminação que, com base nas propriedades do material de cada *voxel* e nas condições de iluminação externas, calcula a tonalidade da cor em cada ponto do volume.

Em algoritmos de extração de superfície aplica-se um dos modelos de iluminação utilizados na Computação Gráfica tradicional; geralmente o modelo de Phong (Phong,1975). Para o *rendering* direto, é comum utilizar-se uma adaptação desse modelo com a substituição das normais à superfície por gradientes do campo escalar no *voxel*.

A última etapa do processo de visualização envolve a projeção dos *voxels*, ou polígonos mapeados, na superfície de visualização e a conseqüente composição para determinar a imagem a ser visualizada. Nas técnicas de Extração de Superfícies, esta etapa realiza a visualização com o auxílio de efeitos de iluminação e remoção de áreas escondidas. Também é implementada nesta etapa a possibilidade de o usuário escolher o formato de visualização mais apropriado, o que pode ser feito através de corte, rotação e especificação do tipo de projeção (ortográfica ou perspectiva). Nos algoritmos de *rendering* direto é realizada a projeção dos *voxels* e o cálculo da composição dos mesmos sobre o plano da imagem.

Neste trabalho procuramos abordar o problema da visualização volumétrica, mostrando suas principais técnicas. No capítulo seguinte são discutidas algumas aplicações de visualização volumétrica, mostrando os formatos dos dados adquiridos e sua dimensão. O Capítulo 3 apresenta uma visão panorâmica das técnicas de visualização volumétrica, incluindo uma perspectiva histórica e uma classificação. O Capítulo 4 apresenta as técnicas de extração de superfícies, dando ênfase ao algoritmo de *marching cubes*. O Capítulo 5 apresenta um modelo que abrange as tarefas básicas da maioria dos métodos de visualização volumétrica direta e discute cada uma das suas etapas básicas com algumas variações. No capítulo 6 são apresentados os principais algoritmos de visualização volumétrica direta, incluindo entre outros os algoritmos de *ray casting*, *shear-warp* e mapeamento de texturas.

2. Aplicações de Visualização Volumétrica

Ainda existem vários desafios no campo da visualização volumétrica. Um dos principais está relacionado ao tamanho dos conjuntos de dados volumétricos, em geral da ordem de vários megabytes. Outro desafio é a necessidade de combinar em uma mesma imagem dois ou mais conjuntos de dados para a análise de algum aspecto comum de interesse.

A visualização volumétrica é vastamente utilizada para comparar dados oriundos de simulações, com resultados numéricos derivados de experimentos empíricos. A análise por elementos finitos ou por dinâmica dos fluidos é utilizada para simular eventos da natureza, constituindo uma fonte geradora de dados volumétricos.

Algumas vezes, a simulação é a única forma possível de estudar um determinado fenômeno. Por exemplo, se um evento é muito grande, muito pequeno, muito rápido ou muito lento para se observar na natureza, então apenas eventos simulados dos dados volumétricos podem ser estudados. Nestes casos a visualização volumétrica passa a ser fundamental no estudo do problema.

Por outro lado, existem dados volumétricos que são obtidos pela digitalização do "objeto de interesse" através de Ressonância Magnética (MRI), Tomografia Computadorizada (CT), Emissão de Pósitron (PET) e Emissão de Fóton (SPECT), ou de técnicas como varredura por laser (microscópicos confocais) e varredura por elétrons (microscópios eletrônicos). Neste caso os dados volumétricos são adquiridos e digitalizados, gerando uma representação do objeto de estudo que se pretende visualizar para estudar algumas propriedades.

Essas variações têm possibilitado que a visualização volumétrica seja amplamente utilizada em medicina, geociências, sensoriamento remoto, meteorologia, astrofísica, química, metalurgia, morfologia, engenharia mecânica e em várias áreas científicas. As grandezas de medida mais comuns a serem visualizadas nestes conjuntos incluem densidade, pressão, temperatura, carga eletrostática e velocidade. Múltiplos valores de cada uma dessas medidas podem ser guardados em um único volume de dados. A seguir são apontados alguns temas de importância em áreas de aplicação ((Câmara,1991) e (Nielson,1994)), bem como as características particulares dos dados.

2.1 Dinâmica de Fluidos

A dinâmica dos fluidos é o estudo do fluxo de fluidos. Este é governado por um conjunto de equações diferenciais denominadas equações de Navier-Stokes, a partir das quais é possível derivar a velocidade e vorticidade do fluxo[1].

Os dados têm a forma $\left(x_{ij}, y_{ij}, z_{ij}; F_{ij}\right)$, com $\left(x_{ij}, y_{ij}, z_{ij}\right) = C\left(u_i, v_j\right)$, onde C é uma representação paramétrica da região de estudo 3D e $u_i, i = 1,\ldots,N_u$ e $v_j, j = 1,\ldots,N_v$ é uma grade retangular. Cientistas utilizam a visualização volumétrica para gerar imagens desses valores tridimensionais, podendo enxergar assim locais de alta vorticidade e compreender melhor o comportamento do sistema.

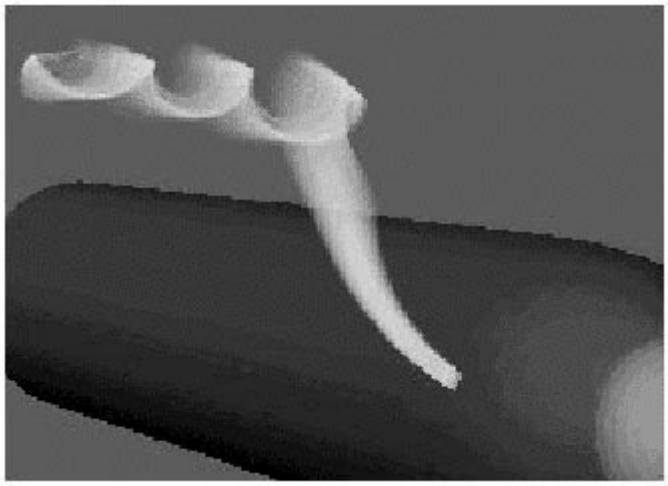


Figura 2.1 Fluxo em torno de um míssil em *grid* curvilíneo

2.2 Medicina

A aplicação de visualização volumétrica a dados médicos tem o objetivo de gerar imagens tridimensionais a partir de fatias bidimensionais (*slices*) oriundas de tomografias, ressonância magnética, ultrassom, medicina nuclear, etc. O objetivo da visualização volumétrica nesta classe de aplicações é auxiliar o profissional a criar uma imagem tridimensional dos dados, facilitando as tarefas de interpretação do diagnóstico, decisão terapêutica ou conduta cirúrgica.

A seguir apresentaremos algumas dessas modalidades de diagnóstico por imagens e algumas aplicações médicas que podem fazer uso destas imagens.

---

[1] Vorticidade está associada a natureza rotacional do fluido.

2.2.1 Métodos de Aquisição de Imagens Médicas

2.2.1.1 Radiografia

Os princípios de aquisição de uma imagem por raios X estão representados na Figura 2.2. Nesta modalidade de aquisição de imagens médicas, um processo elétrico gera um feixe de raios X, que são emitidos isotropicamente seguindo trajetórias retas. A maior parte dos raios que entram no paciente têm sua direção desviada, ou são absorvidos pelo seu corpo. Alguns raios que ultrapassam o corpo do paciente impressionam um filme onde geram uma imagem 2D semitransparente. Este tipo de imagem é excelente para exames rotineiros de pulmões e esqueleto, sendo pouco usado para a visualização de tecidos moles.

Na forma digital, a imagem é representada por um conjunto de dados de 2K *x* 2K *x* 2 bytes, tipicamente com 2 a 4 filmes por exame. No entanto, no caso de angiografia e seriografia digital, são geradas 40 imagens por exame.

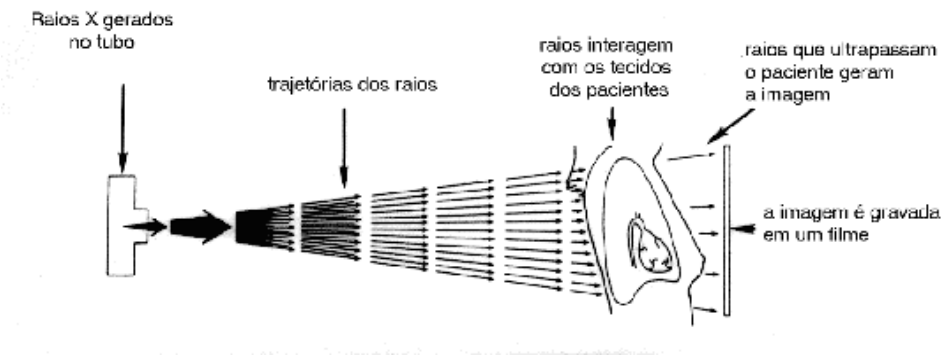


Figura 2.2 Tecnologia de radiografia padrão (raios X)

2.2.1.2 Tomografia Computadorizada (CT)

Em 1972, a Tomografia Computadorizada (CT) revolucionou a área de imagens médicas. A CT é um modo de aquisição de imagens médicas que combina o uso de raios X com tecnologia de computação. Uma série de feixes de raios X é usada partindo de diferentes ângulos para montar imagens de uma seção transversal do paciente. O objetivo é montar essas imagens em um sistema de visualização volumétrica para gerar uma ilustração 3D que possa mostrar órgãos, ossos e tecidos em grande detalhe. A Figura 2.3 apresenta um tomógrafo da Philips.

Figura 2.3 Tomógrafo Tomoscan AV da Philips

Uma imagem de CT é gerada através de um tubo de raios X que gira em torno do eixo do paciente (altura - direção que vai dos pés à cabeça do paciente). Com o auxílio de alta voltagem (120 a 150 KV), o tubo produz um feixe de raios X em leque, que se propaga através do corpo do paciente em níveis controlados. Dependendo da densidade, da espessura e do número atômico dos tecidos do paciente, os raios serão mais ou menos atenuados. Localizados em posição diametralmente oposta ao tubo emissor de raios X estão os detetores. A Figura 2.4 mostra uma representação esquemática destes elementos. Os detetores, em geral de 500 a 1000, estão arranjados na forma de um semi círculo e, dependendo da intensidade de raios X detectada, geram sinais elétricos, que são posteriormente processados por computador para gerar a uma projeção transversal da imagem desejada.

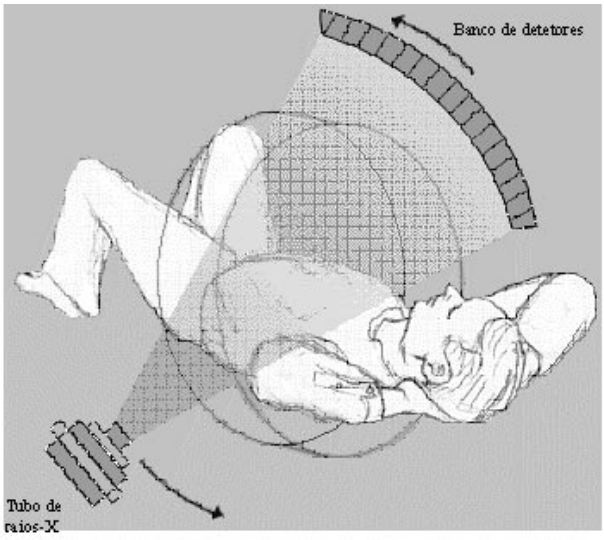


Figura 2.4 Esquema de aquisição da CT

Através da aquisição e do tratamento matemático das inúmeras projeções radiográficas da mesma seção transversal, cada uma adquirida a um ângulo diferente (Figura 2.4), é possível gerar a imagem da seção transversal do paciente. Para atingir este objetivo, a seção transversal é dividida em uma matriz de elementos de volume (*voxels*) com dimensões *d x d x t*. A cada *voxel* é associado um número que depende do valor de medida de alguma propriedade física do tecido. Este esquema está apresentado na Figura 2.5. Para obter os valores em cada um dos pontos desta grade é feita uma reconstrução da imagem através de modelos matemáticos como os apresentados em (Herman,1980) ou (Brooks,1980), que a partir das projeções $P(r,\Theta)$ geram o valor de cada uma das células da matriz que representa a fatia reconstruída do paciente.

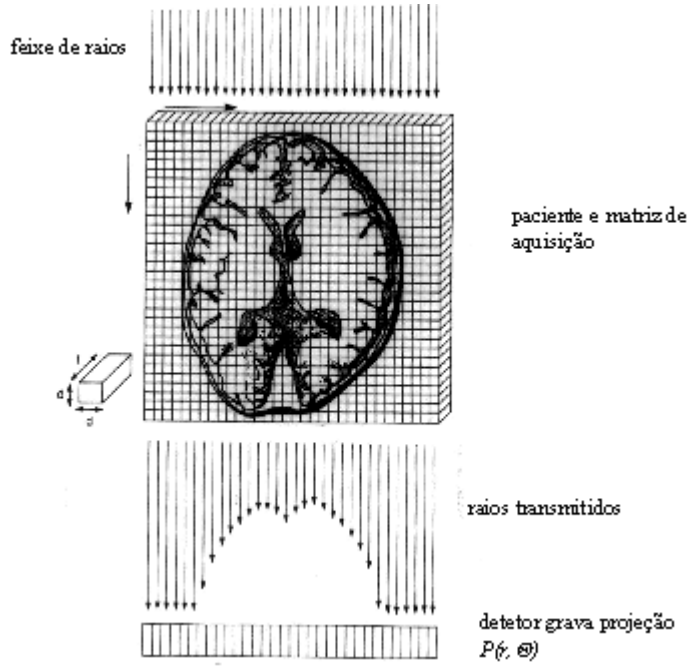


Figura 2.5 Esquema de reconstrução

Transladando o paciente de maneira incremental, transversalmente ao plano de aquisição, gera-se cada uma das fatias que compõem o exame. A translação é de alguns milímetros após cada fatia. Assim, uma série de imagens 2D, igualmente espaçadas, podem descrever estruturas anatômicas 3D com um nível de detalhe submilimétrico.

O resultado de um exame por CT é apresentado como uma série de imagens representando fatias transversais do paciente. Cada fatia representa uma faixa do corpo do paciente com uma espessura entre 1 e 10 milímetros. A maioria dos *scanners*

CT tradicionais geram imagens de 512*x*512 *pixels* para representar cada fatia, com o *pixel* representando porções de 0.5 a 2 milímetros do paciente. Assim, cada *pixel* representa as características de absorção de um pequeno volume em torno de seu ponto central. Em geral são utilizados 2 bytes para representar a intensidade de cada *pixel*.

A intensidade de absorção de cada ponto do corpo do paciente é medida em unidades de Hounsfield (HU). A Tabela 2-1 representa as HU para algumas características anatômicas importantes.

Ar	Gordura	Água	Rim	Pâncreas	Músculo	Fígado	Osso Esponjoso	Osso Compacto
-1000	-110 ±15	0	27 ± 15	35 ±10	40 ±10	55 ±10	200-400	>1000

Tabela 2-1 Absorção de tecidos no exame de CT em HU

Os *scanners* CT podem gerar uma imagem de uma fatia em 1 a 5 segundos, resultando em uma exposição à radiação semelhante às doses da radiografia padrão. Uma tomografia da cabeça leva 10 minutos em média, enquanto a do abdômen gasta em torno de 30 minutos. A Figura 2.6 apresenta dois exemplos de imagens 2D de fatias da cabeça de pacientes.

Em resumo, a CT possui muitas características que a torna mais vantajosa que a radiografia, porém possui seu conjunto de deficiências. As principais deficiências do exame de CT tradicional consistem em:

- pequena resolução temporal para movimento cardíaco;
- presença de artefatos inerentes ao método de aquisição;
- resolução espacial relativamente pequena;
- inabilidade de detecção de doenças em estágios incipientes que não tenham resultado ainda em alterações significantes dos coeficientes de densidade dos tecidos.

Outra modalidade de tomografia computadorizada foi desenvolvida para evitar algumas desvantagens da técnica original: a tomografia helicoidal ou espiral. O termo CT helicoidal é derivado da forma da trajetória percorrida pelo emissor de raios X durante o processo de aquisição. Nesta modalidade, o paciente avança a uma velocidade constante, enquanto o conjunto emissor-receptores gira continuamente, tranversalmente ao eixo de translação do paciente, percorrendo uma trajetória em espiral. A espiral representa um conjunto de dados volumétricos contíguo, que descreve uma porção da anatomia do paciente sem vazios espaciais ou temporais. A

tecnologia de *scanners* CT em espiral permite reduzir a dose de radiação à qual o paciente é exposto, além de resultar na aquisição de imagens com melhor qualidade. Em geral estes exames são 8 a 10 vezes mais rápidos que a CT convencional.

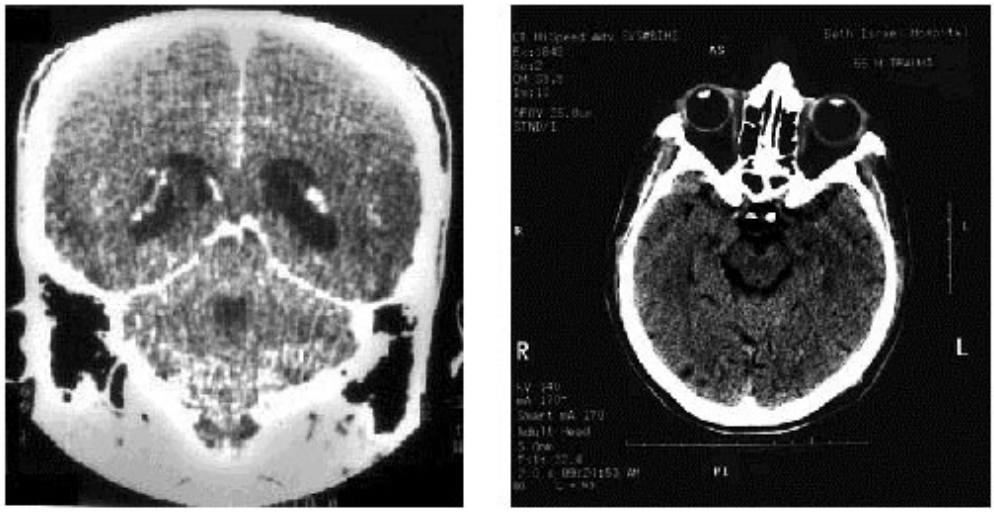


Figura 2.6 Duas imagens de CT da cabeça de um paciente

2.2.1.3 Ressonância Magnética Nuclear (MRI)

Esta modalidade de aquisição utiliza um campo magnético e ondas de rádio para produzir as imagens. Os sinais que as geram dependem de uma série de parâmetros, sendo uma função da abundância de um núcleo específico nas moléculas do tecido amostrado. Dentre os núcleos que geram sinais de MRI, o mais importante em sistemas biológicos é o núcleo de hidrogênio (próton), principalmente devido à abundância de H_2O nos tecidos e ao fato de ser um núcleo estável e bastante sensível à MRI.

Em MRI, o objeto de estudo é colocado dentro de um campo magnético de alta intensidade (Figura 2.7). Isto causa um alinhamento dos momentos magnéticos das moléculas que formam o objeto. Então o objeto é irradiado com pulsos de radiação de microondas de baixo nível (pulsos de excitação), o que faz com que os momentos magnéticos das moléculas oscilem e emitam microondas após cada pulso. Essas emissões secundárias são medidas e armazenadas digitalmente. Através da introdução de gradientes no campo magnético de fundo, é possível determinar a localização espacial de uma onda reemitida. Assim, forma-se uma imagem representando várias

características sobre as emissões das moléculas, em amostras discretas ao longo do objeto de estudo. Através da modificação da freqüência e de características temporais do pulso de excitação e do atraso no tempo antes da realização das medidas da energia emitida, é possível criar imagens de vários tipos particulares de moléculas, movimento e algumas outras características.

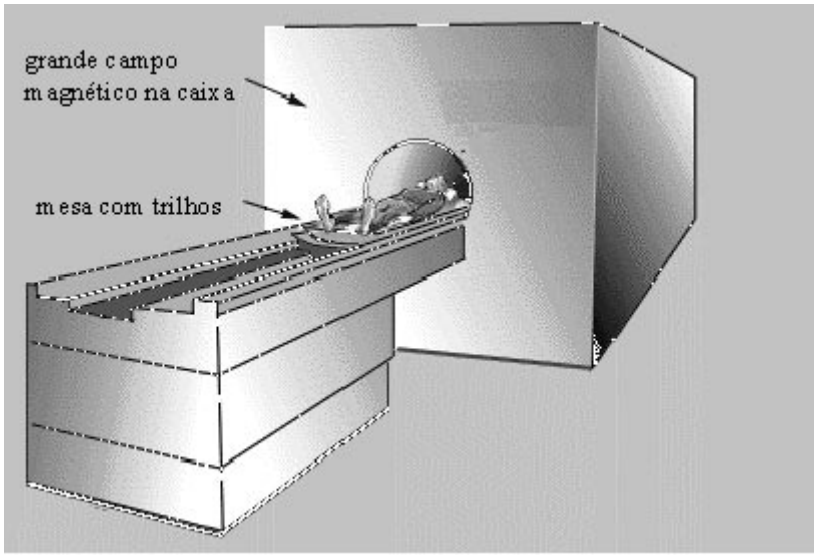


Figura 2.7 Esquema de *scanner* por MRI

As principais vantagens desta modalidade são:

- produção de contraste de tecidos moles superior às outras modalidades, sem a necessidade de agentes de contraste externo;
- detecção de doenças antes do aparecimento de grandes mudanças anatômicas ou fisiológicas;
- obtenção de informações fisiológicas e funcionais;
- as imagens podem ser adquiridas em planos arbitrários, através de manipulação eletrônica, sem a necessidade de mudanças na postura do paciente;
- a ausência de radiação ionizante permite a realização de estudos freqüentes sobre o paciente.

Por outro lado, entre as desvantagens podemos destacar:

- dificuldades no estudo de calcificações;
- suscetibilidade a movimentos do paciente durante a aquisição, por ser um processo lento;

- impossibilidade de aquisição de dados de pacientes em sistemas artificiais de suporte à vida (UTI);
- inexistência de uma escala de valores absolutos para um determinado conjunto de dados.

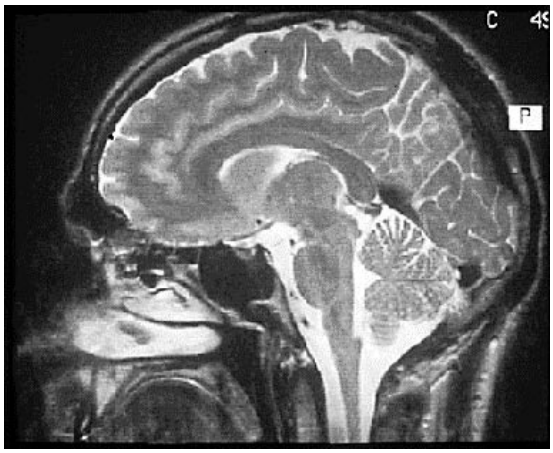


Figura 2.8 Imagem por MRI da cabeça do paciente

O resultado de um exame por MRI é semelhante ao de CT. São apresentadas fatias representando faixas do objeto de estudo, podendo as mesmas estar orientadas em qualquer direção. É comum nesta modalidade de exames submeter o paciente ao exame sob diversos parâmetros para obter a descrição de diferentes características. Em geral as imagens geradas possuem a resolução de 512*x*512 *pixels* com 10 bits para representar cada *pixel*, sendo geradas em torno de 30 fatias por exame. A Figura 2.9 apresenta uma aplicação de MRI funcional (Oliveira Jr.,1997). Nesta aplicação é realizado um exame do cérebro com o paciente sendo submetido a um estímulo; em seguida a atividade cerebral é detectada e comparada (correlação) com o sinal considerado normal, para a identificação de anomalias.

2.2.1.4 Ultra-Som

Os exames de ultra-som utilizam energia acústica para formar uma imagem do objeto de estudo. O princípio de funcionamento é o mesmo do conhecido SONAR (*Sound Navigation and Ranging*), desenvolvido durante a 2ª Guerra Mundial para a detecção de objetos sob a água. Neste exame um pulso sonoro de curta duração e freqüência fixa (entre 1-15 MHz) é emitido através do corpo humano por fontes em contato com a pele. As ondas sonoras se propagam com uma velocidade entre 1450 e 1580 m/s, dependendo do tipo de tecido. A energia é parcialmente refletida ao

encontrar uma interface entre dois tecidos diferentes, sendo detectada por sensores posicionados junto à fonte. As altas amplitudes medidas correspondem a regiões com alta variação de impedância acústica, definida pelo coeficiente de reflexão. Como resultado, o que é visualizado efetivamente é a derivada da distribuição espacial desta grandeza. O modelo do interior do organismo pode então ser reconstruído baseado no tempo de percurso, na velocidade média e na amplitude das ondas refletidas.

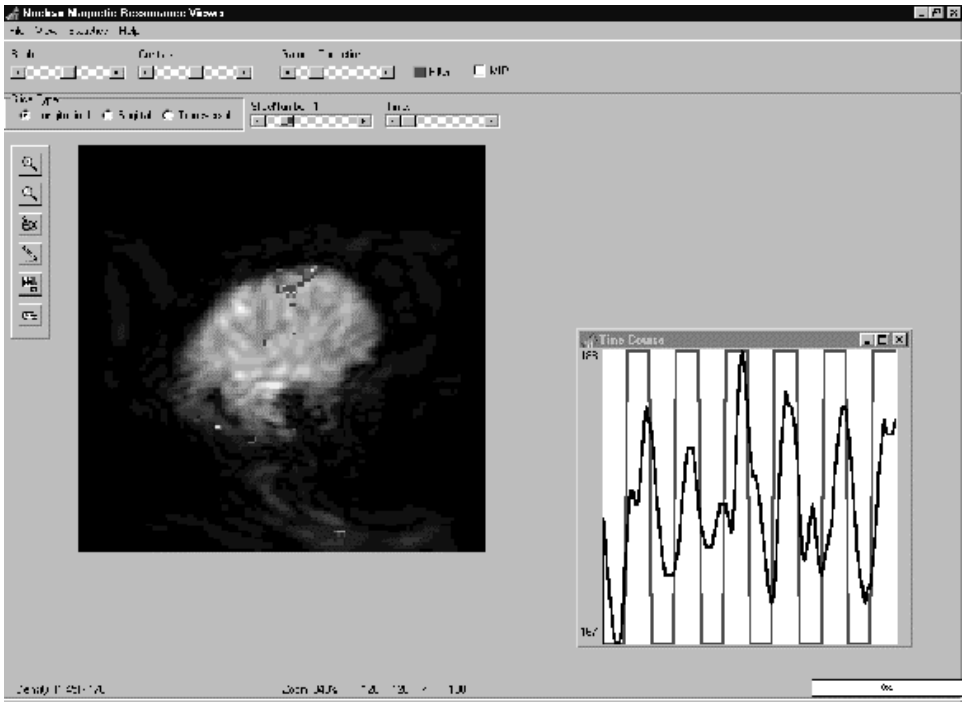


Figura 2.9 Ressonância magnética funcional

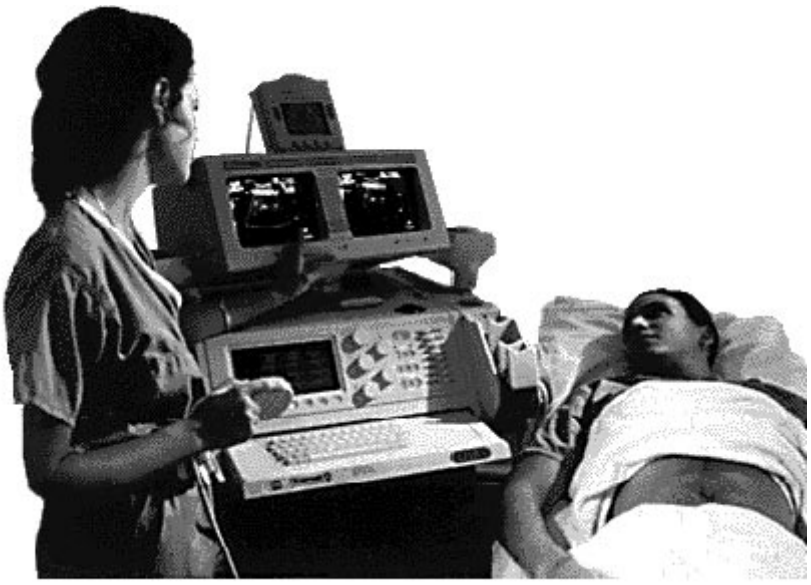


Figura 2.10 Um sistema de aquisição de imagens por ultra-som

Na ultra-sonografia mais comum (bidimensional) tipicamente temos a fonte e o receptor de ondas sonoras montados juntos em um aparelho de mão. O operador coloca-o sobre o paciente e move para obter imagens de várias partes do corpo. A Figura 2.10 representa um sistema de aquisição de dados via ultra-som.

Algumas características dos dados de ultra-sonografia 3D acrescentam dificuldades a visualização, as quais impossibilitam a utilização direta dos métodos de visualização tradicionalmente empregados para CT e MRI (Sakas,1995). Dentre essas as mais importantes são:

1. quantidade significativa de ruídos e *speckle* (ruído não-aleatório resultante de interferência destrutiva incompleta entre energias oriundas de diferentes pontos difusores vizinhos);
2. faixa dinâmica muito menor do que na CT e na MRI (sinal envolvido em um *background* muito ruidoso);
3. grande variação espacial de intensidade, mesmo em regiões de tecido homogêneo;
4. superfícies obscurecidas devido à atenuação da energia em tecidos precedentes nos caminhos dos raios;
5. interfaces com nível de amplitude variável causado pela dependência do ângulo de incidência;
6. as interfaces não são bem localizadas e sim englobam uma região com diversos *voxels*;
7. a amostragem espacial do dado produzido é não homogênea e não previsível;
8. o dado é amostrado em intervalos muito irregulares;
9. este exame sofre dificuldades por não penetrar em estruturas ósseas, ou seja, apenas tecidos moles não envoltos por partes ósseas podem ser registrados.

Entre as características positivas desta modalidade de exames podemos citar:

- não existem evidências consistentes de que as emissões de ondas deste exame sejam prejudiciais;
- o efeito Doppler, produzido pela reflexão de ultra-som a partir de superfícies que se movem, pode ser um ótimo indicador de fluxo

sangüíneo, o que torna este tipo de exame excelente para o diagnóstico de doenças vasculares.

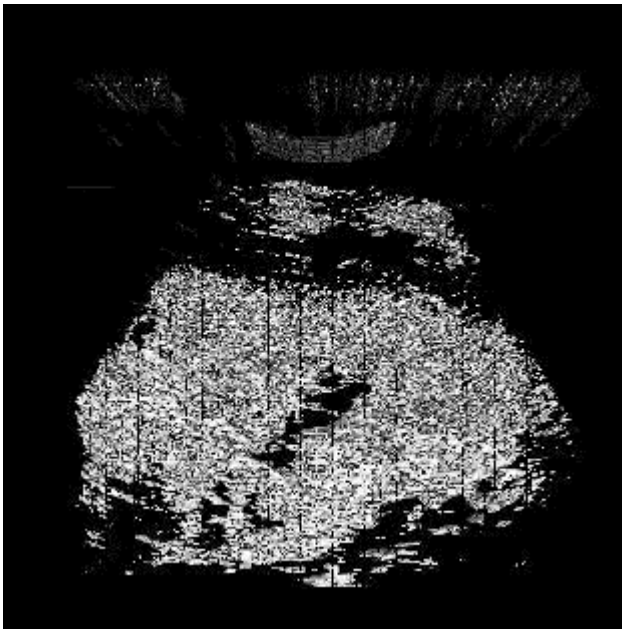


Figura 2.11 Visualização volumétrica de dados de ultra-sonografia 3D

A aparência geral de uma imagem gerada por visualização volumétrica de dados de ultra-sonografia é a de um bloco sólido coberto de um ruído (*speckle)* com aspecto de neve (Figura 2.11). A reduzida faixa dinâmica torna impossível a discriminação entre ruído e informação através de métodos diretos de segmentação (*e.g., thresholding*). As técnicas de visualização baseadas na extração de iso-superfícies falham devido aos itens 1, 3, 4 e 5 listados acima, uma vez que a amplitude acústica nem sempre é representativa da presença de uma interface entre determinados tipos de tecidos e tampouco uma superfície é definida por uma amplitude única.

Em geral, os exames de ultra-som geram imagens de 512*x*512 *pixels* com 1 byte para representar cada *pixel*, utilizando-se 40 imagens por exames. A Figura 2.12 apresenta imagens de exames por ultra-som bidimensional.

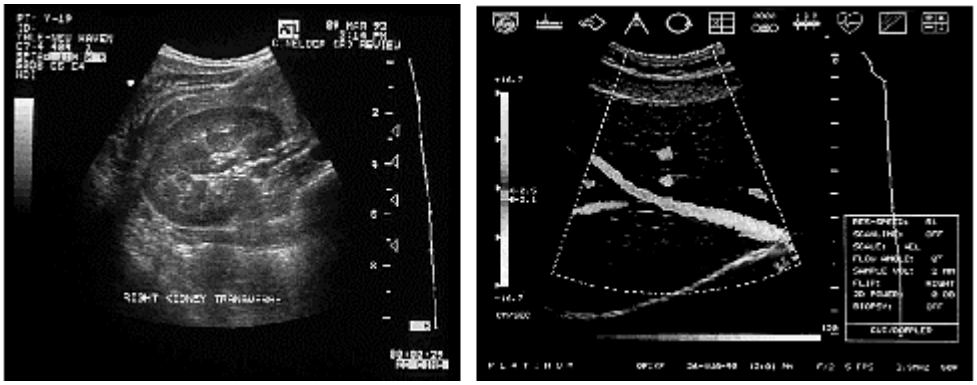


Figura 2.12 Imagens de ultra-som do rim e *dopler* do abdomen

2.2.1.5 Medicina Nuclear

Em exames de medicina nuclear, uma fonte radioativa é injetada no paciente, que é colocado em um detetor (Câmera Gama) para realizar medidas da radiação emitida pelo seu corpo. Através do controle da substância injetada podem ser realizadas imagens de vários aspectos do paciente.

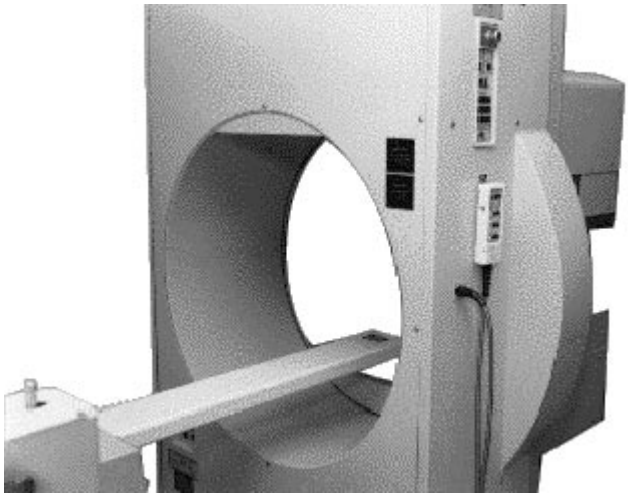


Figura 2.13 Câmera Gama para exames de Medicina Nuclear

A PET é um sistema de aquisição de imagens médicas utilizado para a observação de processos metabólicos no organismo. Este exame possui grande importância em pesquisas e diagnósticos nas especialidades de neurologia, cardiologia e oncologia. Para a realização do exame é administrada no paciente, por injeção ou inalação, uma substância que participa de maneira bem definida no processo bioquímico a ser visualizado. Essa substância é radioativada com isótopos que emitem pósitrons (a antipartícula do elétron) durante o decaimento. O pósitron emitido colide

com elétrons dos tecidos próximos, sendo aniquilado, quando então são liberados dois fótons em direções opostas. Como os fótons possuem alta energia (~511 kV), é grande a probabilidade de sairem do corpo do paciente e serem detectados por uma coleção de detetores localizados ao redor do paciente. Este equipamento gera dados que permitem a obter da distribuição espacial da substância radioativa no corpo do paciente.

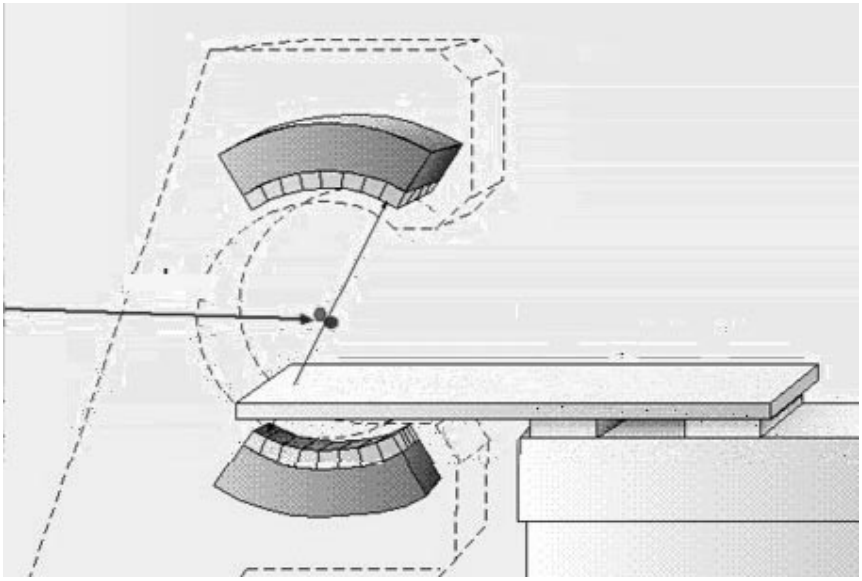


Figura 2.14 Diagrama de aquisição - PET

As imagens geradas são de baixa resolução e com grande quantidade de ruído. Isto se deve à impossibilidade de usar altas taxas de radiação. Os tipos mais comuns são SPECT (*Single Photon Emission Computed Tomography*) e PET (*Positron Emission Tomography*). A SPECT utiliza rastreadores radioativos que emitem fótons enquanto decaem. Os exames com PET utilizam rastreadores radioativos que produzem pares pósitron-elétrons. A PET é mais flexível por ser mais fácil produzir uma grande variedade de substâncias radioativadas. Os resultados desta classe de exames são mais recomendados para a produção de aspectos fisiológicos, os quais podem ou não ser superpostos à anatomia que os contém. Exames de medicina nuclear podem localizar áreas do paciente com atividade funcional alterada, mostrando regiões do organismo que não estejam trabalhando corretamente.

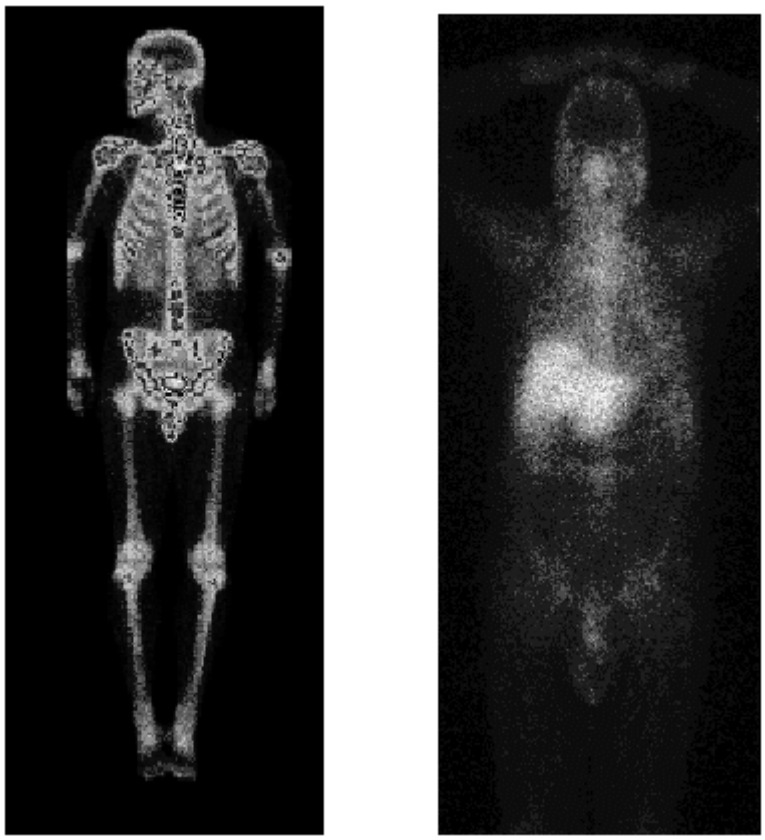


Figura 2.15 Imagens de medicina nuclear

A Figura 2.15 representa duas imagens obtidas através de medicina nuclear. A primeira é uma imagem da estrutura óssea (*bone scan*), usada para a detecção de patologias ósseas o mais cedo possível. Neste exame é injetada no braço do paciente uma substância radioativa que se espalha pela corrente sangüínea, indo se localizar nos ossos. Entre 20 e 60 min após a injeção, coloca-se o paciente na Câmera Gama para a aquisição da imagem. A segunda imagem é resultado de uma aquisição com gálio (*galium scan*), usada para diagnosticar o progresso de tumores ou infecções. Nesta modalidade o período entre a injeção de material radioativo e a aquisição propriamente dita é de 2 a 5 dias.

Os resultados desses exames são tipicamente conjuntos de 10 a 30 fatias transversais, com 256 *x* 256 *pixels* e 2 bytes por *pixel*.

2.2.2 Aplicações

A Tabela 2-2 apresenta um resumo dos exames mais comuns e das características dos dados gerados por cada um deles.

A aplicação primária da visualização volumétrica para a medicina é a área dos sistemas de visualização de exames. Eles devem facilitar o diagnóstico e a identificação de características do paciente examinado.

Exame	Tamanho da imagem	Bytes por pixel	Mbytes Por imagem	Imagens por exame	Mbytes por exame
Raio-X Digital	2K x 2K	2	0.8	2-4	20-40
Seriografia Digital	2K x 2K	2	0.8	40	320
Tomografia Computadorizada	512 x 512	2	0.524	25	13
Ultra-som	512 x 512	1	0.262	40	10
Medicina Nuclear	256 x 256	2	0.131	6	0.8
Ressonância Magnética	512 x 512	2	0.524	30	16

Tabela 2-2 Formato e tamanho dos dados médicos.

Além dessa, outras aplicações se mostram promissoras com a utilização da visualização de volumes.

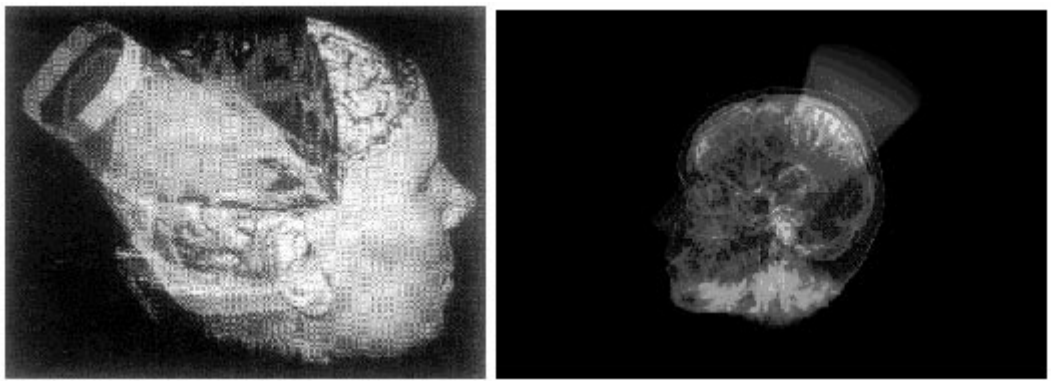


Figura 2.16 Aplicação de radioterapia

Uma delas é a radioterapia, que tem o objetivo de aplicar uma quantidade apropriada de radiação em uma região com um tumor no corpo do paciente. É fundamental que o tumor receba uma carga letal de radiação e os tecidos saudáveis em volta recebam a mínima radiação possível. A visualização volumétrica de um volume digitalizado da região de interesse no corpo do paciente pode ser sobreposta ao equipamento de emissão de radiação, gerando uma simulação que permita ajustar a posição e a dimensão da descarga de radiação. A imagem da Figura 2.16 representa a visualização da cabeça de um paciente com o feixe de radiação da terapia.

Outra aplicação importante é a endoscopia virtual. Ela simula técnicas de inspeção invasivas, que usam um dispositivo ótico introduzido no corpo do paciente

para visualizar as estruturas abdominais. Isto pode ser simulado através do uso de volumes gerados por CT ou MRI, juntamente com técnicas de *fly-trough* para navegar no volume. Esta técnica permite uma visualização 3D da parte interna do abdômen, evitando a desconfortável introdução de um objeto estranho no corpo do paciente.

Na área educacional, podemos citar a construção de atlas digitais criados para a educação de cirurgiões. Pode-se criar um volume de uma região do corpo identificando órgãos e permitindo ao cirurgião explorar as vistas anatômicas.

2.2.3 Interpretação Sísmica

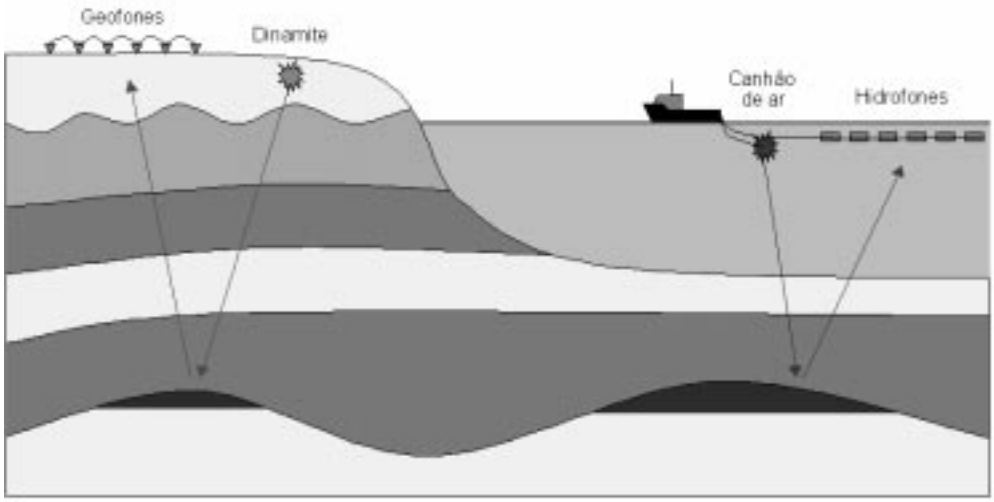


Figura 2.17 Aquisição de dados sísmicos

Uma importante área de aplicação da visualização de volumes é a de exploração de petróleo. Os geofísicos tentam descobrir as estruturas geológicas sob a superfície da terra realizando um trabalho denominado interpretação sísmica.

Os dados podem ser obtidos através da perfuração de um poço e da extração de material para a inspeção das camadas geológicas existentes na área, ou através da aquisição indireta, na qual são detonados explosivos e medidas as ondas refletidas pelas descontinuidades existentes na subsuperfície. As ondas são capturadas na superfície por um conjunto de sensores (geofones ou hidrofones). Este esquema está representado na Figura 2.17.

Os dados obtidos são representados por uma grade 3D de amostras contendo a amplitude das ondas sonoras refletidas abaixo da superfície. As amplitudes são valores reais considerados uniformemente espaçados ao longo dos três eixos. As coordenadas x e y são representadas em metros; a coordenada z, representada em

milisegundos, representa o tempo de propagação e reflexão da onda. Para a representação dos dados sísmicos 3D os valores de amplitude são mapeamento em cor e opacidade. A Figura 2.18 mostra exemplos de volumes sísmicos visualizados. Para uma discussão acerca dos aspectos de visualização volumétrica aplicados à sísmica, ver (Gerhardt,1998).

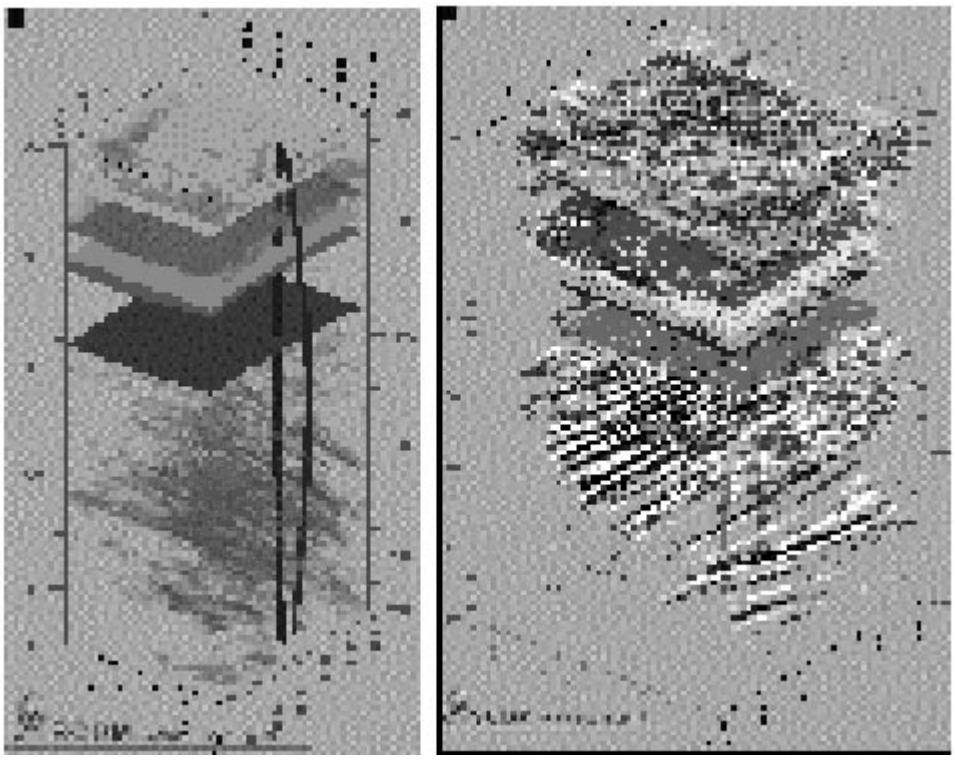


Figura 2.18 Volumes sísmicos com dados volumétricos e de poços.

3. Classificação e Histórico de Visualização Volumétrica

3.1 Introdução

Os algoritmos de visualização volumétrica podem ser descritos em linhas gerais segundos duas classes básicas.

Na primeira classe estão os denominados algoritmos de extração de superfícies (iso-superfície ou *surface fitting*). Esses métodos produzem a visualização através da geração de representações geométricas dos dados de modo a isolar um determinado objeto que estava representado nos dados volumétricos. Em razão disto, reduzem a quantidade de dados manipulados quando da formação da imagem. Esta classe de algoritmos apresenta a necessidade do uso de técnicas de *rendering* de polígonos e de reprocessamento do volume para extrair novamente o objeto toda vez que uma alteração da característica do volume a ser visualizado for solicitada.

Os algoritmos de *rendering* direto constituem a segunda classe. Eles realizam a geram a imagem a ser visualizada diretamente a partir dos dados volumétricos, sem passar por nenhuma representação intermediária. O esforço computacional envolvido é grande, por causa da grande quantidade de dados manipulados durante a geração da imagem. Estes algoritmos permitem a visualização de mais de um objeto contido nos dados volumétricos, tornando possível a visualização de porções interiores aos volumes.

Existem várias formas de caracterizar os diferentes algoritmos existentes para a visualização de volumes. Neste capítulo apresentamos uma discussão acerca das classificações propostas para os algoritmos de visualização volumétrica. Em seguida os apresentamos em uma ordem cronológica, de modo a permitir a criação de uma perspectiva histórica.

3.2 Classificação dos Algoritmos de Visualização Volumétrica

Existem, inicialmente, duas maneiras básicas de propor classificações para os algoritmos de visualização de dados volumétricos. A primeira se baseia no ponto de vista dos dados. Assim podemos classificar as técnicas de acordo com: a natureza dos dados, sua dimensão, objetivos de interpretação, continuidade, etc. (Robertson,1991). Uma outra

forma de realizar esta classificação é sob a ótica das técnicas empregadas no processo de visualização.

Classificações Baseadas nos Dados		
Abordagem de Classificação	Referência	Classes de Técnicas
Variação dos Dados entre Amostras	(Upson,1988) (Elvins,1992)	*voxels*
		células
	(Yagel,1996)	constante
		variável
Arranjo Espacial dos Dados	(Elvins,1992)	regular
		retilíneo
		estruturado
		estruturado em blocos
		não estruturado
		não cartesiano
	(Speray,1990)	cartesiano
		regular
		retilíneo
		estruturado ou curvilíneo
		estruturado em blocos
		não estruturado
		híbrido
Formato de Consideração do Material dos *Voxels*	(Yagel,1996)	opaco
		com atributo de opacidade

Tabela 3.1 Classificações baseadas nos dados

Na primeira categoria de classificações temos as baseadas: na variação do dado entre amostras ((Elvins,1992), (Upson,1988) e (Yagel,1996)), no arranjo espacial dos dados (Elvins,1992) e na forma de consideração do material dos *voxels* (Yagel,1996). Estas classificações estão apresentadas na Tabela 3.1.

Outras classificações são obtidas com base nos procedimentos das técnicas de visualização. Entre elas podemos citar as baseadas: na representação dos dados para *rendering* (Elvins,1992), (Nielson,1995), (Upson,1988), (Coatrieux,1990) e (Yagel,1996) e na ordem em que os dados são considerados ((Elvins,1992), (Yagel,1996) e (Wittenbrink,1998)). Estas classificações estão apresentadas na Tabela 3.2.

Uma classificação apropriada para as diversas técnicas propostas na literatura seria de muita utilidade para a rápida caracterização dos algoritmos. Assim sendo, propomos as classificações apresentadas na Tabela 3.3, as quais são baseadas em uma combinação de atributos das técnicas de visualização. A vantagem da classificação proposta é a de proporcionar uma idéia geral da forma de trabalho do algoritmo.

Classificações Baseadas nos Procedimentos

Abordagem de Classificação	Referência	Classes de Métodos	
Representação Dos Dados	(Elvins,1992)	*Rendering* Direto (DVR)	
		Isosuperfícies (*Surface Fitting*)	
	(Nielson,1995)	Multiplanar Slice Projection	
		Surface Fitting	
		Rendering de Volumes	
	(Upson,1988)	Baseado em Superfícies	
		Baseado em *Voxels*	
	(Coatrieux,1990)	*Rendering* de Superfícies	
		Rendering de Volumes	
	(Yagel,1996)	Visualização Direta de Volumes	
		Rendering de Superfícies	
Ordem de Consideração dos Dados	(Elvins,1992)	Ordem dos Objetos	*Front to Back*
			Back to Front
		Ordem da Imagem	Aleatória
			Scanline
	(Yagel,1996)	*Forward* - Ordem dos Objetos	
		Backward - Ordem da Imagem	
	(Winttenbrink,1998)	*Forward*	
		Backward	
		Forward Multipasso	
		Fourier	

Tabela 3.2 Classificações baseadas nos procedimentos

A classificação por tipos de dados coloca as técnicas de visualização de volumes em duas categorias: Escalar e Vetorial. A distinção entre as duas classes reside no tipo de dado que representa a propriedade do objeto em cada posição *(x,y,z)* do volume. Os métodos que tratam com dados vetoriais visualizam em cada ponto do espaço um vetor, tensor, bimodal ou dado de maior dimensão. Já a classe escalar trata da visualização de um dado unidimensional (escalar) em cada ponto do espaço. Em geral, as aplicações de visualização de volumes está contida nesta classe.

A classificação em função da distribuição espacial dos dados segue a proposta de Speray e Kennon (Speray,1990). Os dados são classificados em sete categorias :

- cartesiano - os elementos de dados são cubos alinhados com os eixos. Este formato é o preferido pela maioria das técnicas de visualização volumétrica porque simplifica o caminhamento no volume;
- regular - os elementos de dados são paralelepípedos de dimensões constantes. Comum em dados provenientes de CT;
- retilíneo - os elementos de dados são paralelepípedos de dimensões variáveis;

- estruturado ou curvilíneo - as células não são lineares. Usados em dinâmica dos fluidos computacional;
- estruturado em blocos - os dados são arranjados em vários grupos estruturados de modo a evitar limitações topológicas;
- não estruturado - sem restrições geométricas, usados em análises do MEF[2];
- híbrido - uma combinação dos formatos anteriores.

Atributos	Valores Possíveis
Tipo do Dado	Vetorial
	Escalar
Distribuição Espacial dos Dados (*Grid)*	cartesiano
	regular
	retilíneo
	estruturado
	estruturado em blocos
	não estruturado
	híbrido
Representação dos Dados	*blocos*
	células
Modelo de Projeção	*Rendering* de Volumes
	Isosuperfícies (*Surface Fitting)*
	Modelos Híbridos
Espaço de Trabalho	Espaco da Imagem (EI)
	Espaço do Objeto (EO)
	Transformações no Espaço do Objeto
	Métodos de Domínio

Tabela 3.3 Classificações baseadas nos atributos das técnicas

Os dados volumétricos são geralmente tratados como uma matriz de elementos de volumes (blocos ou células). Na abordagem por blocos a área ao redor de um ponto do *grid* é considerada como possuindo o mesmo valor do ponto amostrado. Quando os dados são tratados como células o volume é considerado como uma coleção de hexaedros com vértices nos pontos do *grid* e os valores variam dentro da célula através de uma interpolação dos valores dos vértices. Esta abordagem gera imagens mais suaves.

A classificação de técnicas de visualização de volumes mais comum na literatura é a baseada na representação dos dados para *rendering*. Com base nesta abordagem temos as técnicas de visualização através de *rendering* de volumes, extração de superfícies e técnicas híbridas.

[2] MEF - Método dos Elementos Finitos

Os primeiros métodos de visualização de volumes (e.g. *Marching Cubes*) envolviam a aproximação de uma superfície contida nos dados através da visualização de primitivas geométricas. Isto se deu devido ao bom conhecimento das técnicas de exibição (*rendering)* de primitivas geométricas. Entre as desvantagens desta classe de métodos estão a perda de informações durante a conversão e a incapacidade de representar fenômenos amorfos tais como nuvens e fogo. Como representantes desta classe de métodos temos: *Marching Cubes* ((Lorensen,1987) e (Wyvil,1986)), *Contour-Connecting* (Keppel,1975), *Dividing Cubes* (Cline,1988), *Marching Tetrahedras* (Shirley,1990), etc.

As técnicas da classe de *rendering* de volumes projetam o volume diretamente na imagem. Nesta classe encontram-se técnicas como *Ray-Casting* (Levoy,1988), *Splatting* (Westover,1990), *V-Buffer* (Upson,1988) e *Shear-Warping* (Lacroute,1995).

A outra abordagem possível é a que permite a visualização de objetos geométricos juntamente com objetos representados por *voxels*. Alguns métodos realizam a conversão para uma das representações, outros não. Entre as abordagens que não realizam a conversão estão a baseada em pontos (Johnson,1989) e as que suportam modelos de dados híbridos ((Kaufman,1990), (Goodsell,1989) e (Levoy,1990c)). Nesta classe de técnicas podemos citar *Z-Merging* (Duff,1985) e *Ray-Merging* (Goodsell,1989).

A classificação em função do espaço de trabalho só faz sentido para técnicas de *rendering* de volumes. Os métodos do espaço do objeto calculam a contribuição de cada *voxel* nos valores dos *pixels* da imagem; assim, a combinação das contribuições de todos os *voxels* resulta na imagem final. Nos métodos do espaço da imagem, são lançados raios a partir dos *pixels* da tela em direção ao volume, calculando as contribuições de cada *voxel* ao longo do raio para formar a cor do *pixel*. Existem alguns algoritmos que realizam parte do processo de visualização no espaço dos objetos, para em seguida realizar as operações no espaço da imagem. A estes métodos denominamos baseados em transformações no espaço do objeto. Nesta classe de métodos podemos citar o algoritmo proposto por Drebin (Drebin,1988) e o *Shear-Warping* apresentado em (Lacroute,1994). Nos métodos de domínio, os dados espaciais são transformados para um domínio alternativo, tal como compressão (Yeo,1995), *wavelet* (Muraki,1993) ou domínio da freqüência (Malzbender,1993), a partir do qual a projeção é gerada diretamente. A Tabela

3.4 representa as principais técnicas classificadas segundo atributos de modelo de projeção e espaço de trabalho.

Algoritmos de Visualização de Volumes					
Extração de Superfícies	*Rendering* Direto de Volumes (DR)				Híbridos
	Ordem dos Objetos	Ordem da Imagem	Transf. no Espaço do Objeto	Métodos de Domínio	
Cubos Opacos (*Cuberrille)* (Herman,1979)	V-Buffer (Upson,1988)	Ray Casting (Tuy,1984)	Volume Rendering (Drebin,1988)	Fourier (Dunne,1990)	Ray-Merging (Goodsell,1989) (Levoy,1990)
Contour Connecting (Keppel,1975)	Splatting (Westover,1990)	Cell Integration	Shear-Warping (Lacroute,1994)	Fourier (Malzbender, 1991)	Z-Merging [Duff,1985]
Marching Cubes (Lorensen,1987)	Transf. do Volume (Hanharan,1990)	Método de Sabella (Sabella,1988)		Wavelet (Muraki,1993)	
Dividing Cubes (Cline,1988)	Mapeamento Texturas (Wilson,1994)			DCT (Yeo,1995)	
Marching Tetrahedra (Shirley,1990)	Projeção Coerente (Wilhelms,1991)				

Tabela 3.4 Classificação de técnicas de visualização de volumes

A Figura 3.1 apresenta uma ilustração do aparecimento das técnicas de visualização de volumes.

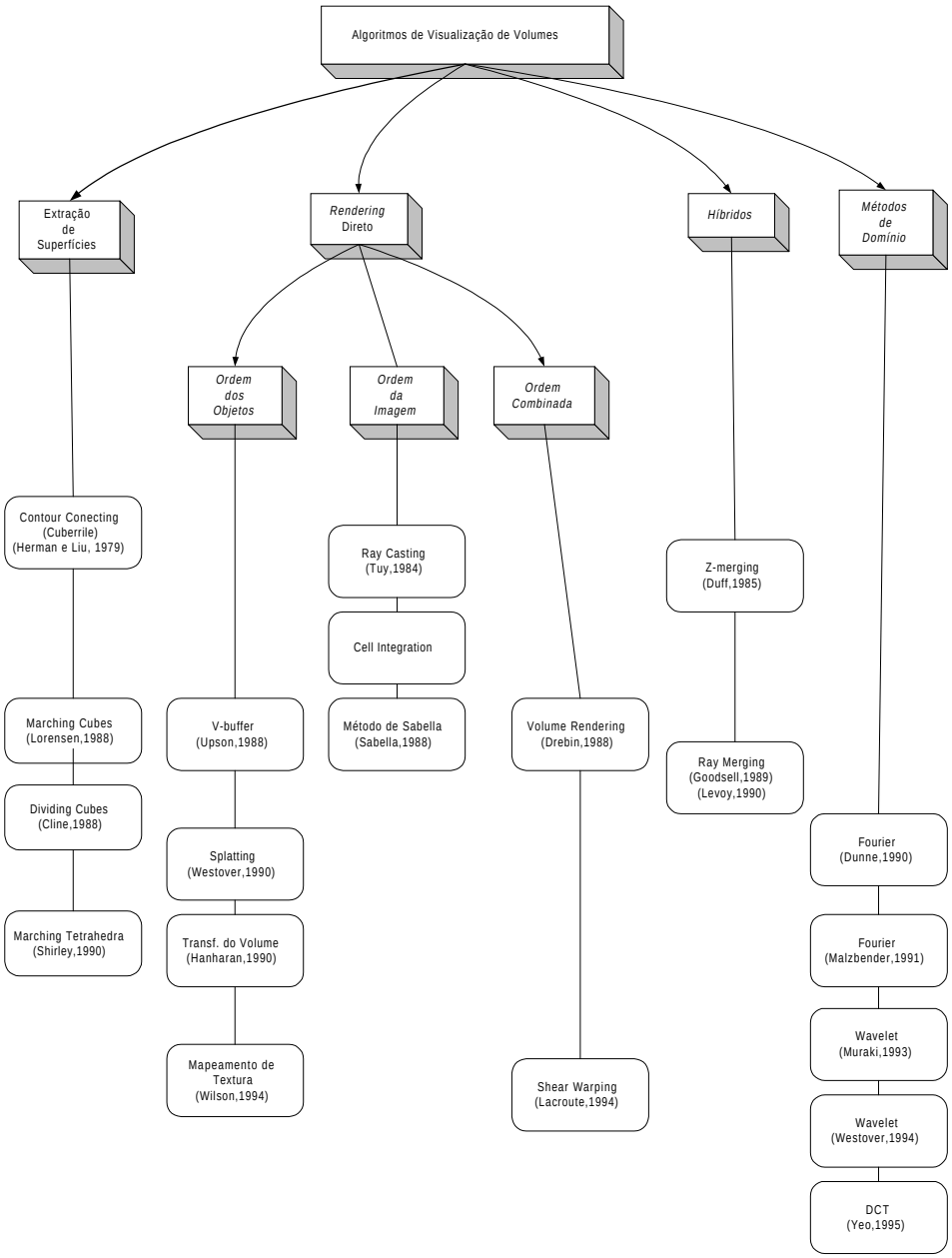


Figura 3.1 Perspectiva histórica das técnicas de visualização de volumes

4. Algoritmos de Extração de Superfícies (SF)

Os algoritmos de extração de superfícies tipicamente ajustam uma superfície, discretizada em polígonos, a pontos de isovalor dentro dos dados volumétricos. O processo se inicia com a escolha por parte do usuário de um valor de limiarização (*thresholding*). Então, primitivas geométricas são ajustadas automaticamente aos pontos por onde deveria passar a superfície de valor especificado. Após a definição da isosuperfície são utilizados os métodos tradicionais de *rendering* de polígonos para realizar a visualização do volume; isto possibilita a utilização de métodos rápidos já conhecidos, assim como de *hardware* especificamente desenvolvido para este fim.

Uma desvantagem do uso de isosuperfícies está no fato de visualizarmos somente um subconjunto dos dados, determinado pelo valor de limiarização. Isto pode ser contornado com o uso de isosuperfícies transparentes, uma para cada valor de *limiarização*. No entanto, a compreensão da cena diminui à medida que incluímos mais isosuperfícies. Em geral estes algoritmos geram um grande número de primitivas geométricas, o que pode se tornar um problema para a visualização.

Outro problema sério desses algoritmos é a inclusão errônea de artefatos na imagem, resultantes de classificações erradas da superfície (e.g. devido a problemas de aproximação) e a dificuldade de visualizar detalhes do volume. A geração de isosuperfícies é a forma mais antiga e mais bem conhecida de se estudar o conteúdo dos conjuntos de dados volumétricos. No final dos anos 70, Fuchs (Fuchs,1977) desenvolveu um algoritmo para gerar representações geométricas tridimensionais através da conexão das isolinhas de planos adjacentes.

Posteriormente, Lorensen e Cline (Lorensen,1987) apresentaram um algoritmo denominado *Marching Cubes* que gera uma isosuperfície examinando os oito vértices do *voxel* e determinando as superfícies de interseção. As interseções ao longo das arestas do *voxel* são calculadas usando-se interpolação linear ou trilinear e, então, é gerada uma malha triangular para a isosuperfície. O algoritmo de *Dividing Cubes* funciona da mesma forma, porém a isosuperfície é gerada por pontos, ao invés de triângulos, através de subdivisões sucessivas.

4.1 Contour-Connecting

A idéia básica de se traçar uma isolinha em cada fatia (*slice*) de dados e depois conectar fatias adjacentes foi inicialmente sugerida por Keppel (Keppel,1975) e posteriormente refinada por Fuchs *et all* (Fuchs,1977) e por Ekoule (Ekoule,1991).

Contour-Connecting é um método de detecção de superfícies (SF), com caminhamento pelos objetos (ordem dos objetos), que opera inicialmente em cada fatia de dados individualmente. Após o usuário especificar um valor de limiarização (*threshold*), uma curva fechada conecta estes valores para cada fatia de dados (linha de isovalores).

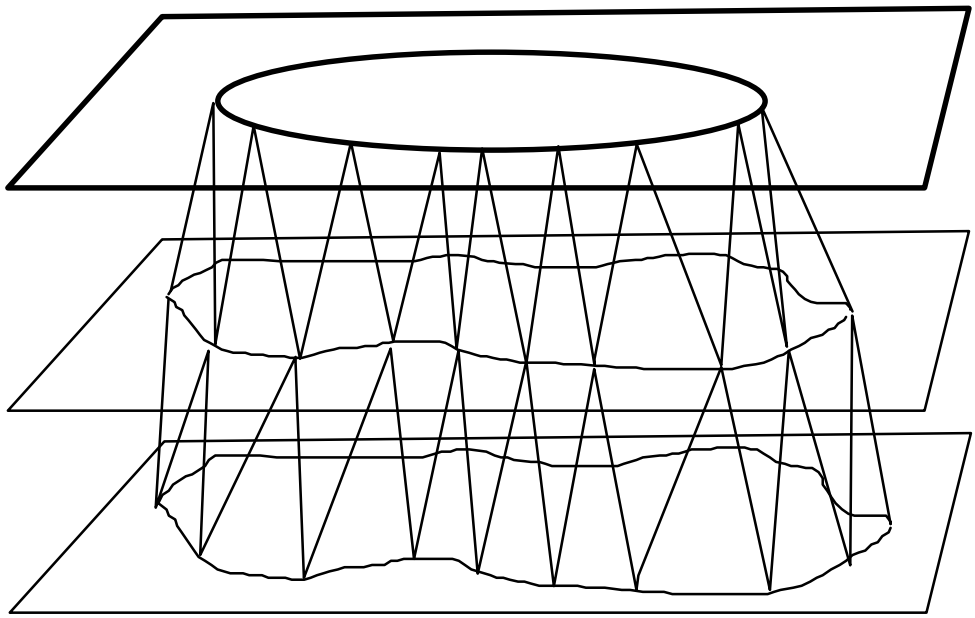


Figura 4.1 Algoritmo *Contour Conecting*

Uma vez determinado o contorno de cada fatia, o problema torna-se encontrar a conexão ótima, geralmente através de triângulos, interligando as curvas de fatias adjacentes (Figura 4.1). Sendo este o problema central, principalmente quando uma fatia contém mais de um contorno que pode ser conectado a um dos contornos da fatia seguinte, em geral é usado um critério de proximidade (explícita ou implicitamente) para definir os pares de contornos a serem conectados. A Figura 4.2 ilustra esta dificuldade.

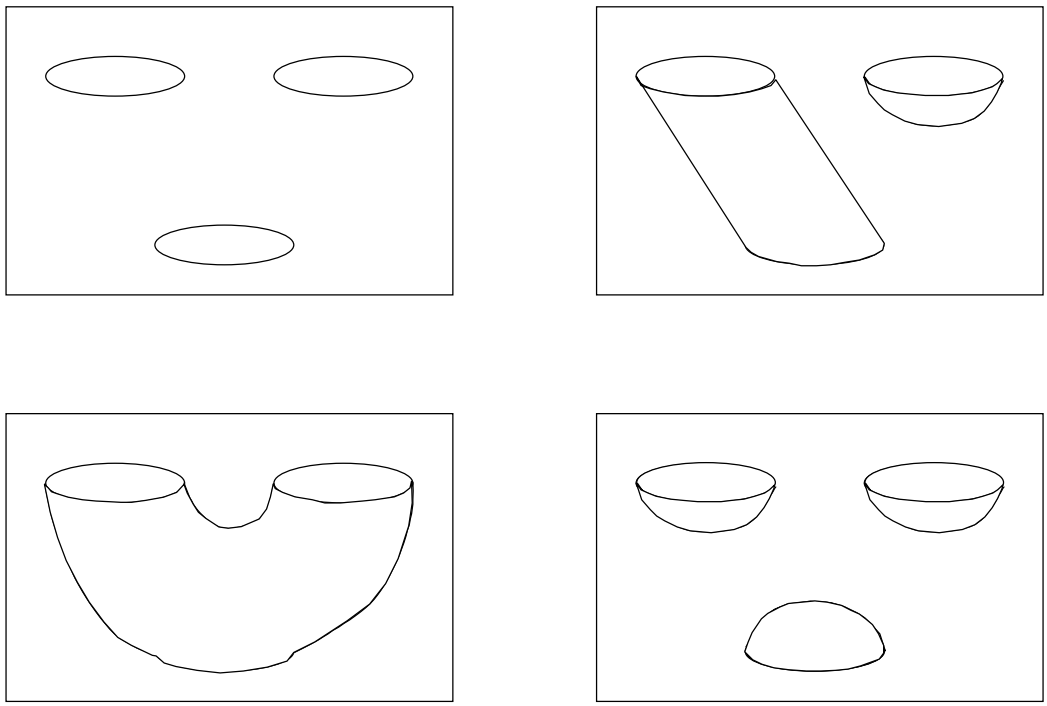


Figura 4.2 Modelos que podem ser gerados para um par de fatias

Este algoritmo tenta tirar proveito da abundância de métodos de visualização de superfícies conhecidos. A parte do algoritmo de detecção de superfícies é paralelizável, uma vez que a detecção das isolinhas em uma fatia é completamente independente das outras fatias.

A seguir trataremos do problema de conectar dois contornos já definidos em duas fatias adjacentes, sem entrarmos em detalhes sobre o tratamento do problema apresentado na Figura 4.2

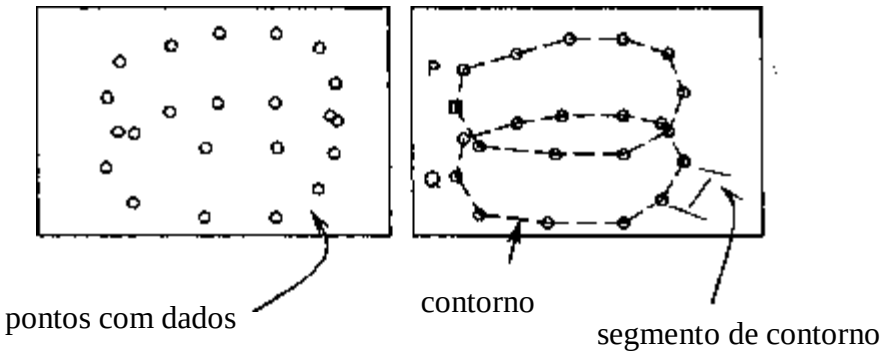


Figura 4.3 Pontos estruturados formando contornos

Consideremos um contorno como um ciclo fechado de segmentos de reta ligando pontos de mesma propriedade em um plano particular, sendo todos os planos paralelos entre si (Figura 4.3). Consideremos dois contornos situados em planos adjacentes, definidos como: $P_o, P_1, P_2 \cdots P_{m-1}, P_o$ e $Q_o, Q_1, Q_2 \cdots Q_{n-1}, Q_o$.

Existem dois problemas básicos para a construção de uma superfície a partir de conjuntos estruturados de contornos planos: correspondência e acoplamento. O problema da correspondência ocorre quando um segmento de um contorno no plano é criado e existem dois ou mais contornos próximos no mesmo plano. Ou seja, o problema é: "como construir um contorno dado um conjunto de pontos em um plano". Algumas soluções para este problema foram sugeridas por Meyers *et all* (Meyers,1992). Uma solução emprega cilindros elípticos, que produzem contornos ajustados por elipses; outra solução usa árvores geradoras de grafos que ligam os contornos em seções adjacentes, o grafo gerado é usado para encontrar a melhor conexão dos contornos.

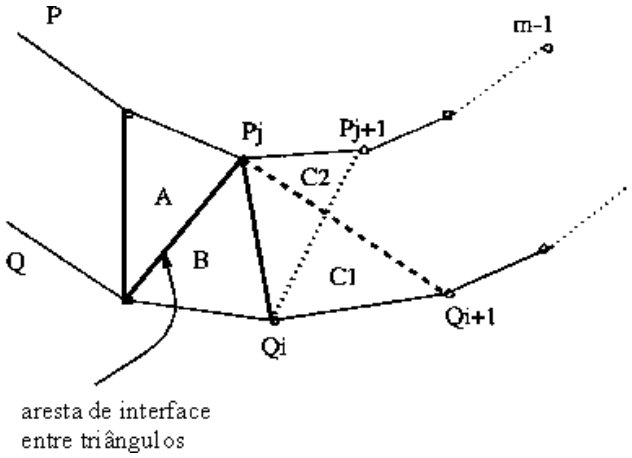


Figura 4.4 Dois contornos paralelos

O problema do acoplamento está relacionado à maneira como os contornos serão unidos em duas fatias adjacentes para formar os triângulos que geram a superfície. Cada triângulo é formado por dois pontos do contorno P e um do contorno Q, ou vice versa. Isto pode ser escrito como: (P_i, P_j, Q_k) *ou* (Q_i, Q_k, P_j). O problema é decidir a orientação do próximo triângulo (Figura 4.4), se ele será C_1 ou C_2. Fuchs (Fuchs,1977) determina o triângulo seguinte utilizando as seguintes regras:

- cada segmento de contorno será usado por um único triângulo;
- o lado esquerdo de um determinado triângulo será usado como lado direito do seguinte.

Para resolver o problema, foi utilizada a teoria de grafos. A decisão do próximo triângulo a ser conectado é feita com base na seguinte hipótese: “Toda superfície aceitável definida entre dois contornos pode ser associada com certos ciclos em um grafo toroidal direcionado”. A superfície ótima corresponde ao caminho mínimo no grafo toroidal. O custo é a soma dos arco percorridos. A Figura 4.5 apresenta uma representação toroidal e o grafo associado; a Figura 4.6 mostra um grafo direcionado gerado para um segmento de contorno e a triangulação gerada. Várias heurísticas são propostas para calcular este caminho mínimo.

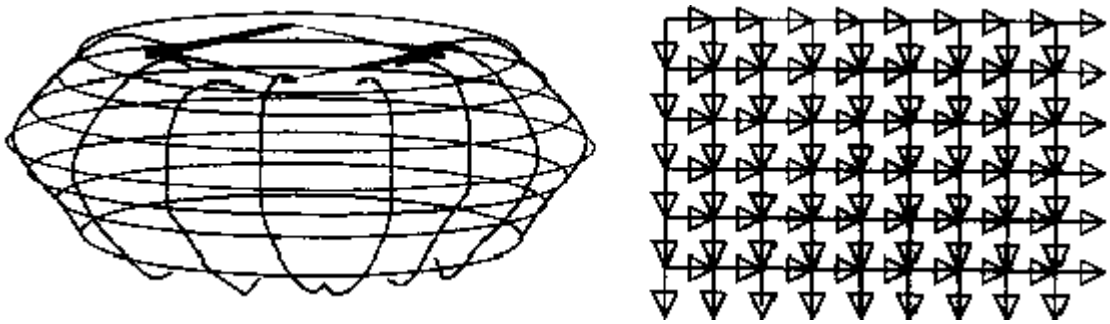


Figura 4.5 Representação toroidal e grafo dirigido associado

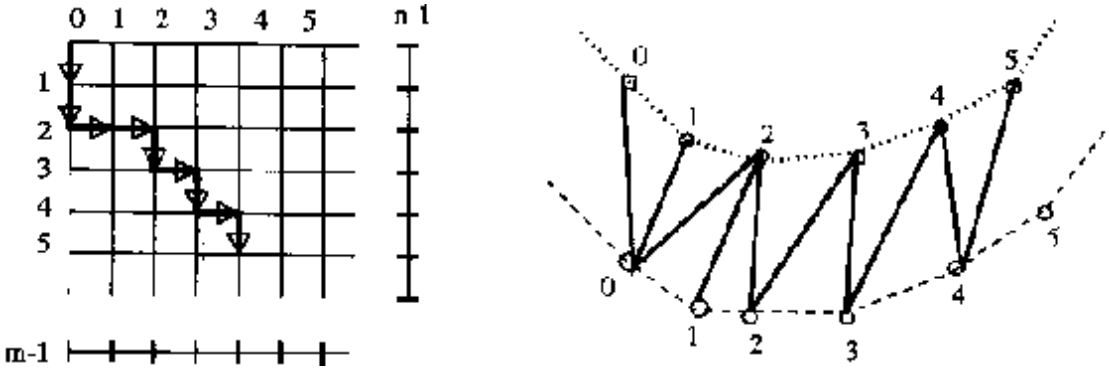


Figura 4.6 Grafo com contorno marcado e trecho de contorno triangularizado

4.2 Opaque Cube ou Cuberille

Esta técnica foi originalmente proposta em (Herman, 1979) e depois aprimorada e otimizada por vários outros autores.

O algoritmo é executado em duas etapas. Na primeira etapa o usuário determina um valor de limiarização (*threshold*). O volume é então percorrido procurando-se as células cujos valores nos vértices encontram-se no nível de *threshold*. Seis polígonos, um para cada face da célula, são gerados para estas células. A segunda etapa transfere a descrição geométrica destes polígonos a um visualizador de superfícies para a geração final da imagem (Figura 4.7). Se múltiplos valores de *thresholds* forem escolhidos, cada conjunto correspondente de células é visualizado com uma cor e opacidade específica para ser diferenciado dos outros.

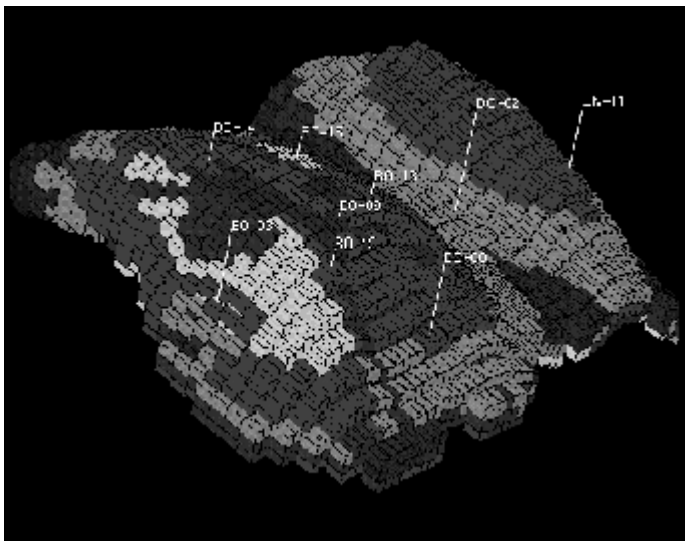


Figura 4.7 Imagem gerada com o algoritmo *Opaque Cubes*

Cubos opacos ou semi-opacos são usados para representar o contorno de superfícies de isovalor (isosuperfícies), causando na superfície a aparência de blocos na imagem final. Esta aparência de blocos pode ser reduzida se utilizarmos algoritmos de visualização que levam em conta o gradiente das superfícies.

Apesar do algoritmo *opaque cubes* ser fácil de implementar e rápido, ele possui a mesma deficiência da maioria dos algoritmos de extração de superfícies: a visualização de detalhes pequenos. A Figura 4.8 apresenta a visualização de dados de reservatórios juntamente com dados de poços em uma aplicação na indústria de petróleo.

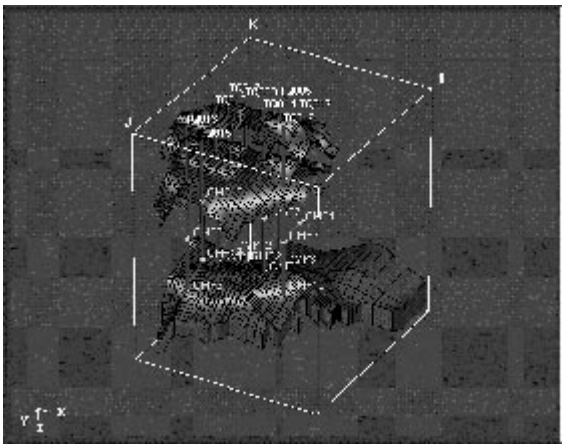


Figura 4.8 Aplicação do algoritmo *Opaque Cubes*

Uma característica importante neste algoritmo é o fato da primeira etapa poder ser paralelizada, uma vez que todas as células podem ser testadas e os polígonos gerados independentemente.

4.3 Marching Cubes, Dividing Cubes e Outras Técnicas

O algoritmo de *marching cubes* foi desenvolvido independentemente por Wyvill e Mcpheeters (Wyvill,1986) e por Lorensen e Cline (Lorensen,1987). Wyvill utilizou este algoritmo para extrair isosuperfícies de um campo analítico definido por um conjunto de pontos. Já Lorensen aplicou a idéia à criação de modelos de superfícies de densidade constante, a partir de dados médicos 3D resultantes de exames de Tomografia Computadorizada (CT), Ressonância Magnética (MRI), etc.

O algoritmo apresentado por Lorensen se baseia em dois passos básicos:

- localização da superfície correspondente ao valor especificado pelo usuário e geração dos triângulos que aproximam esta superfície;
- cálculo das normais nos vértices dos triângulos.

Lorensen e Cline (Lorensen,1988) descobriram posteriormente, que o tamanho de alguns triângulos gerados era menor que o tamanho de um *pixel*. Um novo algoritmo, denominado *dividing cube*, foi então desenvolvido para tirar vantagem desta observação. O algoritmo efetua a projeção das células na tela verificando somente aquelas em que a projeção é maior que um *pixel*. Se isso ocorrer, a célula é dividida em sub-células, cada uma das quais determinando um ponto. Se não, a célula toda é visualizada também como um ponto.

4.3.1 O Algoritmo Básico de *Marching Cubes*

O algoritmo *marching cubes* é uma das técnicas utilizadas para visualizar dados amostrados, uma classe de dados que ocorre com freqüência em áreas como Tomografia Computadorizada, simulações numéricas em dinâmica de fluidos, etc.

O problema de visualização de dados amostrados pode ser abstratamente definido por:

"Seja uma função f(x,y,z), tal que $F:[0,1]^3 \rightarrow [0,1]$, onde [0,1] é o intervalo unitário sobre o conjunto dos números reais. Seja N um número natural e $\Delta = 1/(N-1)$. Uma amostragem uniforme de f pode ser definida como o vetor tridimensional F de N^3 pontos tal que para i,j,k = 1,2,...,N-1 , $F(i,j,k) = f(i\Delta, j\Delta, k\Delta)$" .

O problema de visualização de dados amostrados pode ser caracterizado então como o problema de construir uma aproximação poligonal discreta S, de uma isosuperfície I definida por $f(x,y,z) = \alpha$.

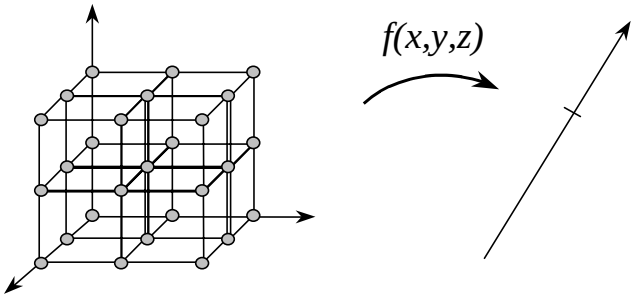


Figura 4.9 Problema de visualização de dados amostrados

Um algoritmo para a solução deste problema deve possuir características como:

- fornecer uma superfície contínua;
- aproximar uma isosuperfície que seja uma função contínua dos dados de entrada;
- gerar uma isosuperfície com topologia correta, que seja invariante com respeito aos sinais dos valores amostrados;
- não adicionar à isosuperfície características que não possui;

- gerar um número pequeno de polígonos, de ordem *O(M)*, onde *M* é o número de células que interceptam a isosuperfície.

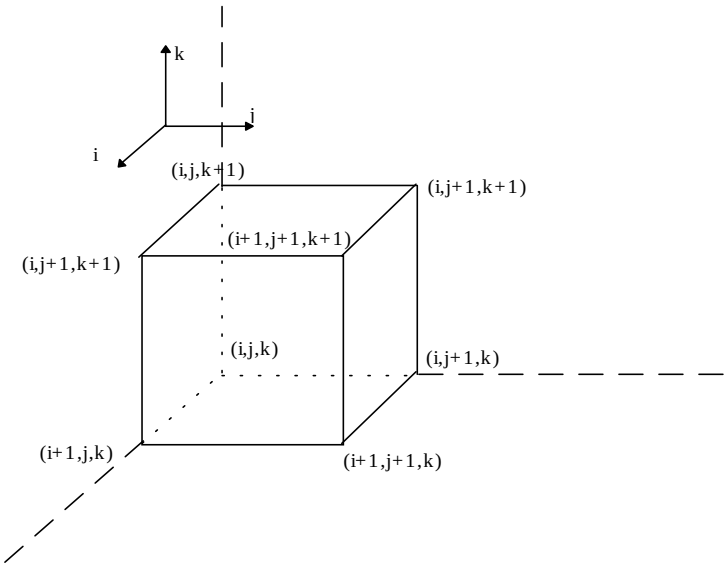


Figura 4.10 Cubo na posição (i, j, k)

Para determinar a superfície no volume, consideramos os cubos formados por cada conjunto de 8 nós adjacentes do *grid* de pontos amostrados. Assim, temos que o cubo localizado na posição (i,j,k) é formado pelos vértices do *grid* localizados nas posições: {*(i,j,k), (i+1,j,k), (i+1,j+1,k), (i,j+1,k), (i,j,k+1), (i+1,j,k+1), (i+1,j+1,k+1), (i,j+1,k+1)*}. Isto está ilustrado na Figura 4.10.

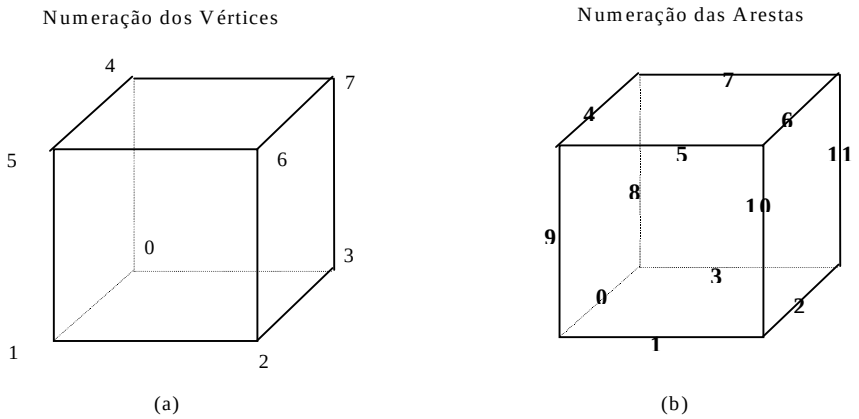


Figura 4.11 Numeração dos vértices e das arestas

Para cada um dos vértices avalia-se o seu valor com relação ao da superfície, classificando-o. Se atribuirmos 1 aos vértices de valor superior ao da superfície e zero aos

vértices de valor inferior, poderemos codificar os cubos. O código de cada cubo é dado pela expressão: $cod = \sum_{i=0}^{7} 2^i * val_i$, onde val_i é o valor do vértice $i \in \{0,1\}$, e i é o número do vértice conforme mostrado na Figura 4.11a. De acordo com o código do cubo temos a definição da topologia da superfície dentro dele.

O Algoritmo 4-1 representa a idéia básica.

```
inicio
  para i de 1 até MAX_I
    para j de 1 até MAX_J
       para k de 1 até MAX_K
          [1] calcula o código do cubo em (i,j,k)
          [2] de acordo com o código do cubo calculas as interseções
              da superfície com as arestas
          [3] gera os triângulos conforme o código do cubo
       fim para
    fim para
  fim para
fim
```

Algoritmo 4-1 *Marching Cubes*

O *loop* central do algoritmo é feito de tal modo que a ordem na qual os cubos são considerados (caminhamento no *grid*) se dá conforme ilustrado na Figura 4.12. Primeiro consideramos os cubos por fatias do volume; depois, dentro de cada fatia trata-se linha a linha.

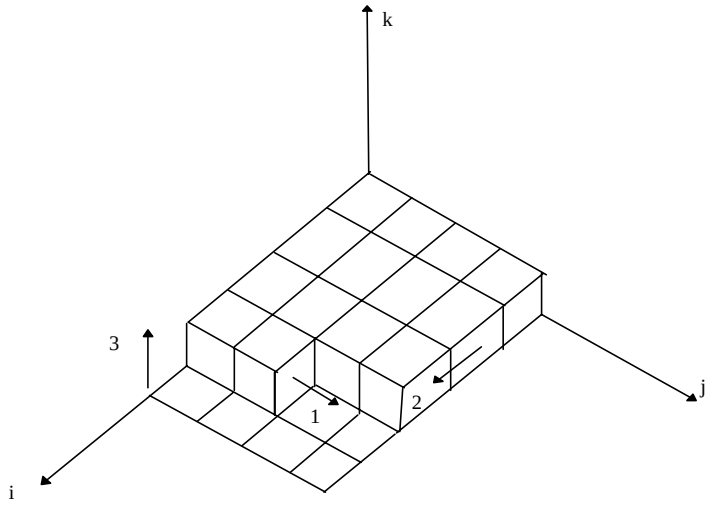


Figura 4.12 Ordem de caminhamento no espaço dos cubos

Um ponto central ao algoritmo é a definição da tabela que determina a topologia da superfície dentro do cubo para cada caso possível. No trabalho de Lorensen, os 256 casos da tabela são reduzidos a 15 casos básicos, através de considerações de rotação e complementaridade. A primeira consideração parte do princípio de que a topologia da superfície é invariante quanto a rotações do cubo e a segunda baseia-se no fato do caso complementar, obtido invertendo-se os valores dos vértices, ser equivalente ao caso original. Os 15 casos básicos propostos por Lorensen são apresentados na Figura 4.13.

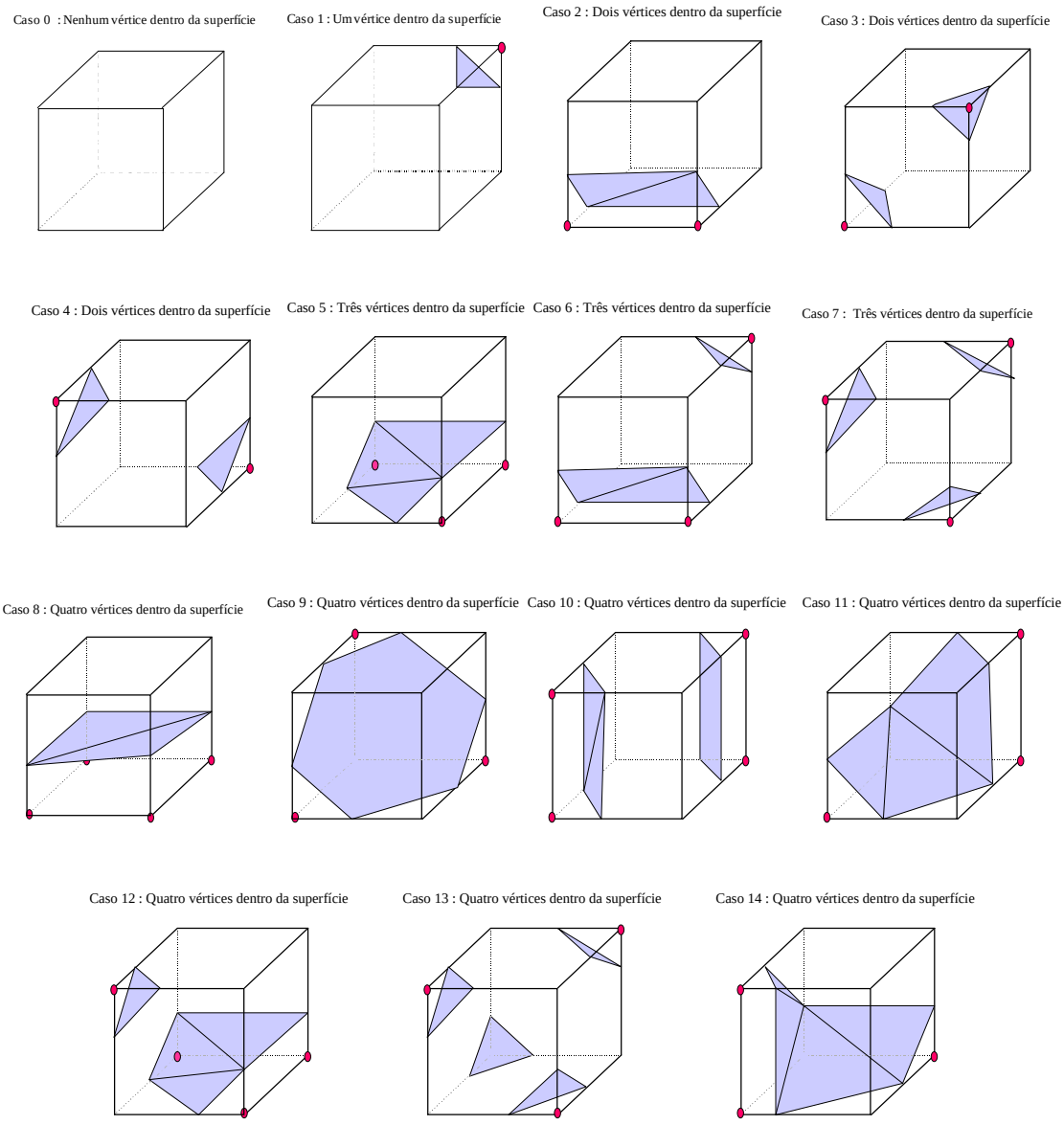


Figura 4.13 Casos básicos de Lorensen

A partir dos 15 casos básicos, deve-se gerar a tabela com os 256 casos. Isto pode ser feito com o seguinte algoritmo:

```
início
  para cada caso básico (0 a 14)
    para cada rotação possível
        calcule o índice_do_cubo
        realize a rotação das arestas interceptadas
        escreva as interseções na tabela na posição índice_do_cubo
        escreva as interseções na posição 256-índice_do_cubo
    fim para
  fim para
fim
```

Algoritmo 4-2 Criação da tabela de *Marching Cubes*

Podemos detalhar o algoritmo como:

```
início
  lê pontos da primeira fatia em fatia_B
  para k=1 até MAX_K
    fatia_A ← fatia_B
    lê pontos da fatia k em fatia_B
    para i=1 até MAX_I
      para j=1 até MAX_J
        ind = indice do cubo (i,j,k)
        GeraTriangulos (ind, i,j,k)
        Imprime Faces
      fim para
    fim para
  fim para
fim
```

Algoritmo 4-3 Detalhamento da criação da tabela de *Marching Cubes*

A função **GeraTriangulos**, baseada no índice do cubo e nos dados da tabela, determina que arestas são interceptadas pela superfície, calcula as interseções e gera as faces triangularizadas.

Para uma implementação eficiente é necessário que esta rotina aproveite a coerência espacial, evitando calcular um ponto de interseção superfície-aresta mais de uma vez. Assim, quando um ponto de interseção é calculado, ele é armazenado em uma tabela e referenciado pelo índice. Além disso são usados alguns vetores para armazenar os índices de pontos já calculados, de modo que possam ser reutilizados, quando necessário, em passos subseqüentes.

As arestas de um determinado cubo devem ser processadas de modo a minimizar o número de *buffers* necessários. A Figura 4.14 mostra os *buffers* utilizados: xedge[MAX_I][MAX_J], yedge[MAX_I][MAX_J], zedge[MAX_I], topyedge[MAX_I]. Os *buffers* xedge e yedge permitem o aproveitamento da coerência fatia a fatia. Os outros *buffers* permitem o aproveitamento da coerência linha a linha. Além desses *buffers*, usam-se duas variáveis *old10pt* e *old6pt* para aproveitar a coerência *pixel* a *pixel*. As arestas devem ser processadas segundo a ordem: 0 4 6 3 7 5 8 9 10.

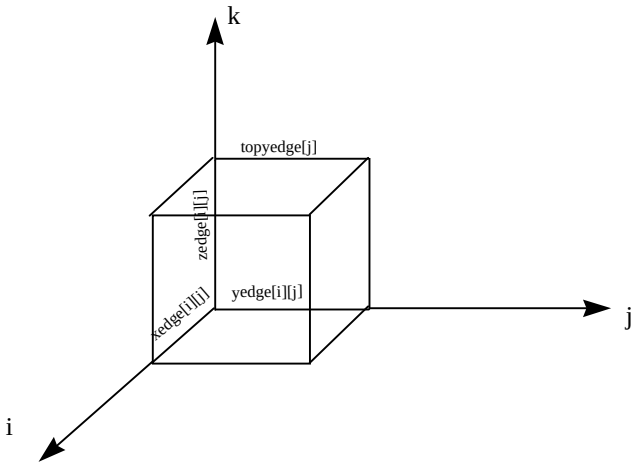


Figura 4.14 *Buffers* utilizados durante a execução do algoritmo

O algoritmo de processamento das arestas para a geração da superfície é dado por:

```
Inicio
  enquanto próxima_aresta do cubo é válida
    para próxima_aresta igual a:
          :0  vertice = xedge[NJ*i+j];
          :1  vertice = yedge[NJ*(i+1)+j]
          :2  vertice = xedge[NJ+i+j+1]
          :3  vertice = yedge[NJ*i+j]
          :4  vertice = old6pt //  6 do anterior
              xedge[NJ*i+j] = vertice
          :5  vertice = cálcula interseção
              inclui vertice na tabela
              topyedge[j] = vertice
              se é o último cubo da linha
                  yedge[NJ*(NI-1)+j] = vertice
              fim se
          :6  vertice = cálculo da interseção
              inclui vertice na tabela
              old6pt = vertice
              se é a última linha
                 xedge[NJ*i+NI-1] = vertice
            fim se
```

```
          :7  vertice =topyedge[j]
              yedge[NJ*i+j] = vertice
          :8  vertice = zedge[j]
          :9 vertice = old10pt // 10 anterior
              zedge[j] = vertice
          :10 vertice = calcula interseção
                inclui vertice na tabela
                old10pt = vertice
                se é o último cubo da linha
                    zedge[NI-1] = vertice
                fim se
          :11 vertice =zedge[j+1]
    fim para
    edge_table[aresta]= vertice
  fim enquanto
  adiciona novas faces ao modelo
fim
```

Algoritmo 4-4 Processamento de um cubo para gerar a superfície

Conforme apontado por Dürst (Dürst,1988) o algoritmo de *marching cubes* apresenta um problema básico, o que pode levar à geração de isosuperfícies com descontinuidades (furos). Um exemplo deste problema ocorre quando um cubo do caso 6 é adjacente a um cubo do caso complementar ao 3, conforme mostra a Figura 4.15:

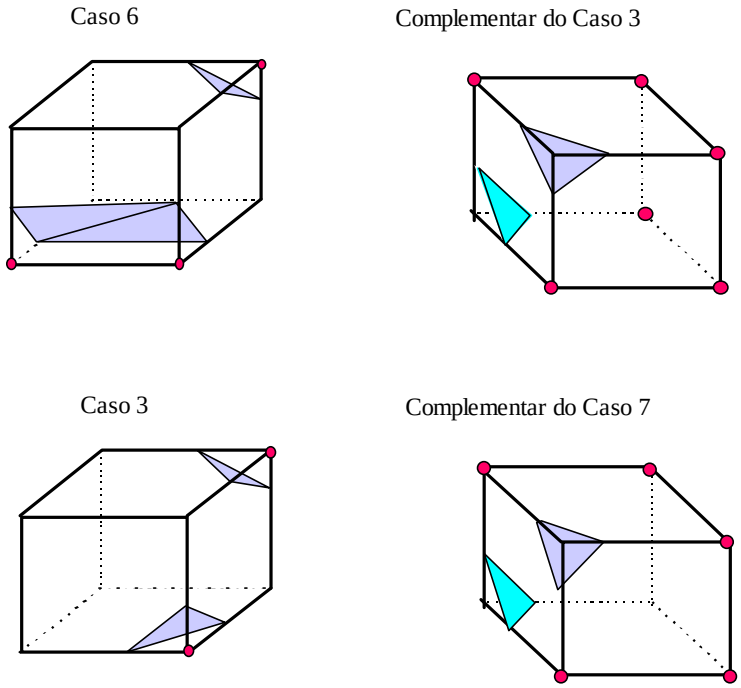


Figura 4.15 Aparecimento de descontinuidades

Este problema de geração de isosuperfícies sem continuidade C^0 deve-se ao fato dessa superfície ser indeterminada em certos cubos, se levarmos em consideração apenas os valores dos vértices. Quando isto acontece apresenta-se uma isosuperfície com topologia ambígua.

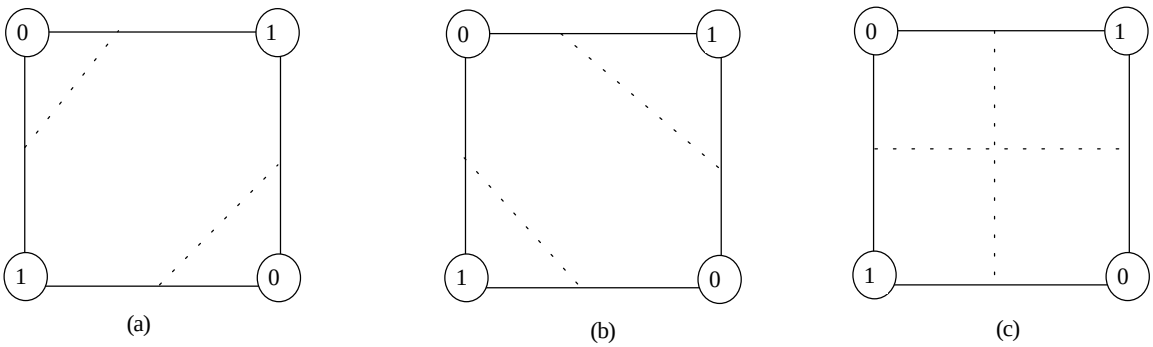


Figura 4.16 Ambigüidade topológica em face 2D

A principal manifestação desta ambigüidade ocorre quando em uma determinada face aparecem dois vértices positivos e dois vértices negativos, posicionados de forma oposta pela diagonal. Neste caso a isosuperfície intercepta as quatro arestas. Conforme mostra a Figura 4.16, não é possível, somente a partir do sinal dos vértices, determinar qual configuração é a correta.

Um outro caso pode ocorrer quando dois vértices positivos estão opostos por uma das diagonais do cubo. Para esta configuração a isosuperfície pode ser um tubo ou dois triângulos. Neste caso, mesmo que não haja a geração de descontinuidades, pode-se gerar uma isosuperfície que não corresponda à realidade, ou seja, que possua uma topologia incorreta.

Existe um grande número de abordagens para escolher uma topologia particular em casos ambíguos. A seguir apresentaremos algumas delas.

4.3.2 Métodos de Eliminação de Ambigüidades

4.3.2.1 Métodos Booleanos Simples

Esta é a categoria dos métodos mais simples. Como faces triangulares não podem ser ambíguas, um método para eliminar as ambigüidades implicitamente, baseia-se na decomposição de cada célula (cubo) em tetraedros e na consideração de variação linear no tetraedro ((Koide,1986), (Bloomenthal,1988) e (Doi,1991)). A divisão neste método não é isotrópica, requerendo uma escolha arbitrária que influencia a topologia resultante.

Nesta classe também está incluído o método de *Marching Cubes* Modificado. Nele uma tabela com 22-23 casos pode ser construída a partir da consideração dos casos complementares como casos distintos, garantindo a compatibilidade nas triangularizações.

Udupa e Ajjanagade (Udupa,1990) descrevem três abordagens que podem ser usadas na construção dessa tabela modificada:

- sempre conectar as diagonais negativas ((Lorensen,1987), (Baker,1989) e (Kalvin,1991));

- sempre conectar as diagonais positivas ((Artzy,1981), (Herman,1983), (Chen,1985));
- conectar as diagonais positivas se a face está em um plano, de outro modo conectar as diagonais negativas.

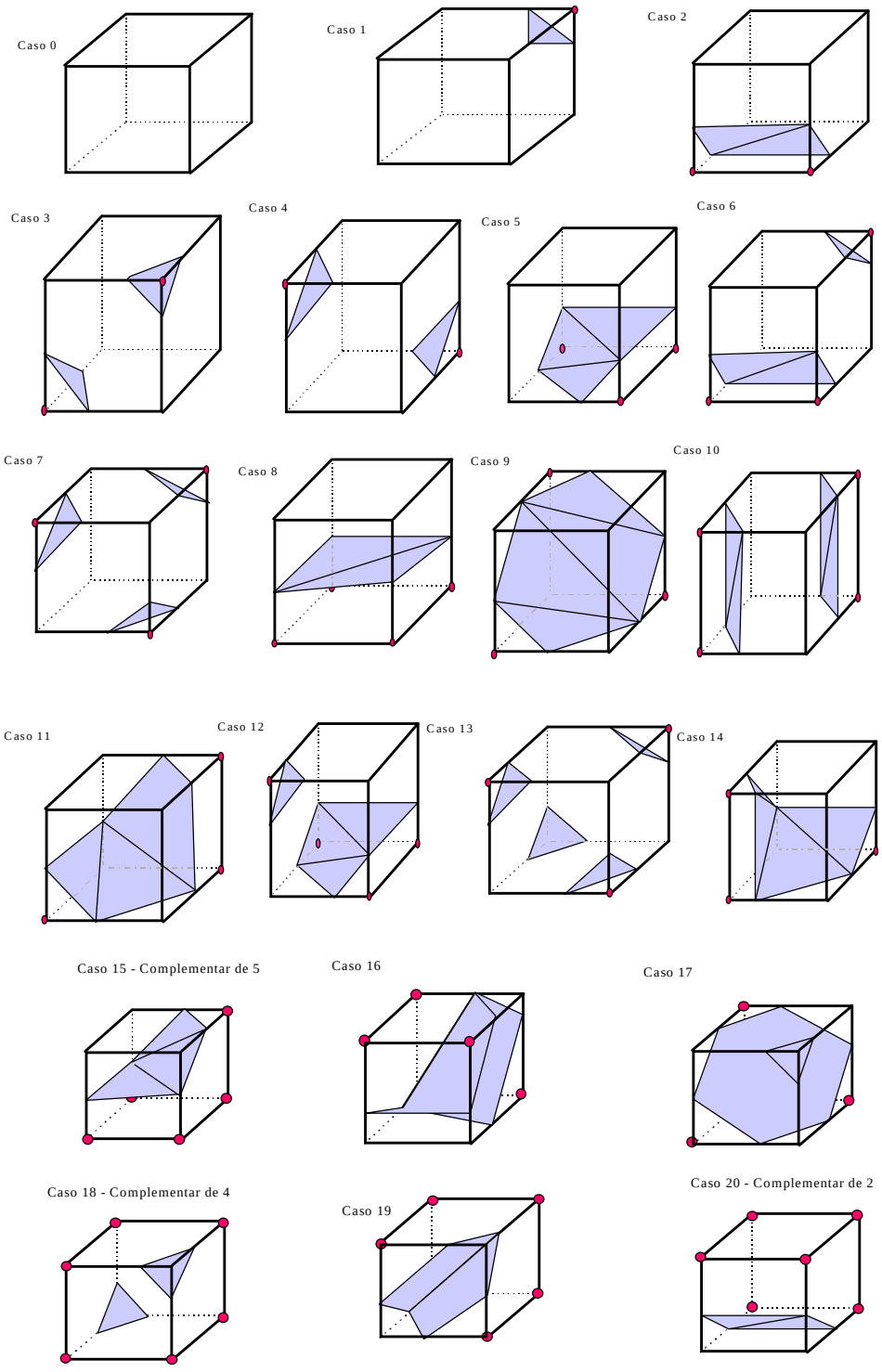


Figura 4.17 Tabela modificada de *Marching Cubes*

A proposta que corresponde à primeira abordagem foi desenvolvida independentemente por ((Baker,1989) e (Kalvin,1991), (Natarajan,1991), (Nielson,1991)). Nesta proposta, as células com 5 ou mais vértices positivos são tratadas independentemente de seu caso complementar e sempre tendo os vértices negativos de uma face ambígua conectados. Este método garante a continuidade da superfície gerada, mas não garante que sua topologia seja a correta. A Figura 4.17 apresenta uma tabela modificada.

4.3.2.2 Métodos Métricos Simples

Nesta classe de métodos encontram-se o método do valor médio na face e os modelos bilineares.

O método do valor médio na face, apresentado em (Wyvill,1986), usa a média dos 4 valores dos vértices em uma face ambígua para eliminar a ambigüidade. Por utilizar apenas os valores da face, este método pode gerar superfícies com a topologia incorreta, mesmo garantindo a continuidade.

Na estimativa do comportamento de dados amostrados, para extrair isosuperfícies (Cline,1988), para realização de *rendering* direto de volumes ((Levoy,1988) e (Upson,1988)), e para refinar a amostragem em uma determinada região, utiliza-se o método de interpolação trilinear.

Nielson e Hamann (Nielson,1991) consideraram modelos bilineares em cada face para tomar decisões de conectividade.

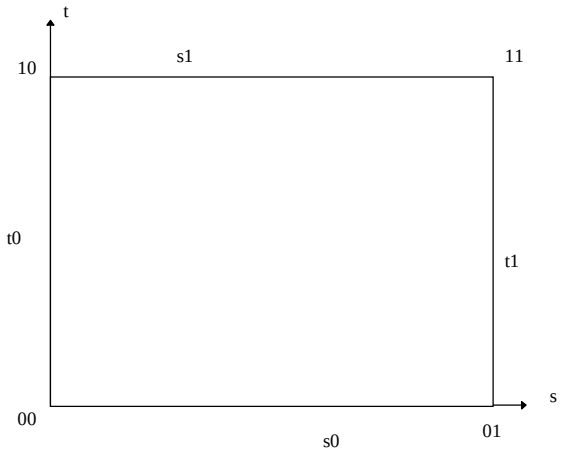


Figura 4.18 Sistema local de uma face do cubo

Considerando um sistema local na face (Figura 4.18), identificando os vértices por {00,01,10,11} e assumindo que os valores dos vértices são dados por $\{f_{00}, f_{01}, f_{10}, f_{11}\}$, podemos definir a interpolação bilinear nesta face como:

$$I(s,t) = (1-s, s)\begin{bmatrix} f_{00} & f_{01} \\ f_{10} & f_{11} \end{bmatrix}\begin{Bmatrix} 1-t \\ t \end{Bmatrix}$$

Se considerarmos um caso ambíguo em uma face, com $I(s,t) = \alpha$, podemos eliminar a ambigüidade comparando o valor α com o valor de $I(s,t)$ no ponto assintótico da face (s_a, t_a).

O ponto (s_a, t_a) é o ponto onde $\partial I / \partial s = \partial I / \partial t = 0$. Assim temos:

$$s_a = \frac{f_{00} - f_{01}}{f_{00} + f_{11} - f_{01} - f_{10}}, \quad \text{e} \quad t_a = \frac{f_{00} - f_{10}}{f_{00} + f_{11} - f_{01} - f_{10}}.$$

O valor $I(s_a, t_a)$ é dado por:

$$I(s,t) = \frac{f_{00} f_{11} - f_{10} f_{01}}{f_{00} + f_{11} - f_{10} - f_{01}}$$

A ligação dos vértices é feita da seguinte maneira:

```
se α > I(s_a, t_a) então
        conecte (s1,1) a (1,t1)
        conecte (s0,0) a (0,t0)
senão
        conecte (s1,1) a (0,t0)
        conecte (s0,0) a (1,t1)
fim se
```

Algoritmo 4-5 Eliminação de Ambigüidade em Face Plana 2D.

Assim como o método do valor médio, que considera somente os valores na face ambígua, este método pode resultar em uma topologia incorreta.

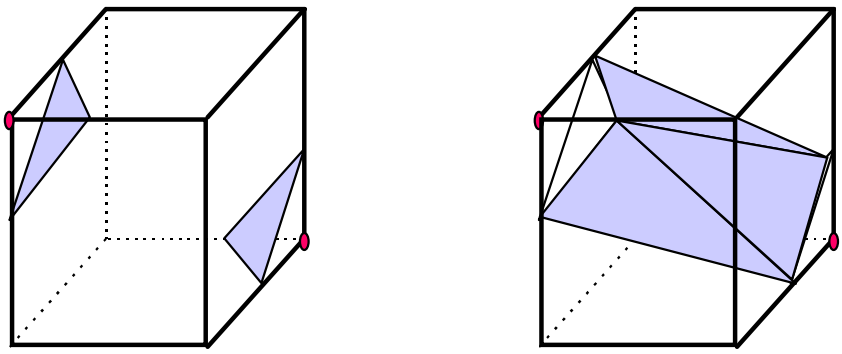


Figura 4.19 Ambigüidade do caso 4

Ainda nesta classe de métodos, Natarajan (Natarajan,1991) apresenta um método baseado em interpolação trilinear, que resolve inclusive os problemas de ambigüidade mais raros, como o do caso 4 da tabela. Neste caso a dúvida se dá entre uma superfície interpolada com dois triângulos ou um tubo passando pelos vértices positivos (Figura 4.19).

Em seu trabalho, Natarajan acrescenta as novas triangulações surgidas destas ambigüidades, como subcasos da tabela original de *Marching Cubes*, e decide que topologia escolher com base no valor *I(s,t,u)* da interpolação trilinear no ponto assintótico no cubo. Este método é análogo ao modelo bilinear, diferindo apenas por tratar a interpolação trilinear no cubo, e não a interpolação bilinear na face ambígua.

5. Rendering de Volumes

As técnicas de *rendering* de volumes geram uma imagem do volume diretamente a partir dos dados volumétricos, sem a necessidade de representações intermediárias. Estes algoritmos são especialmente apropriados para a visualização de volumes que representam objetos amorfos, tais como nuvens, gás, fluidos, etc.

Outra vantagem dos métodos diretos de *rendering* de volumes reside no fato desses métodos separarem as operações de classificação das operações de iluminação. Isto permite que a aparente orientação da superfície, obtida na iluminação, seja independente de erros de classificação.

Entre as desvantagens destes algoritmos podemos citar a necessidade de percorrer todo o volume a cada vez que uma imagem precisar ser gerada.

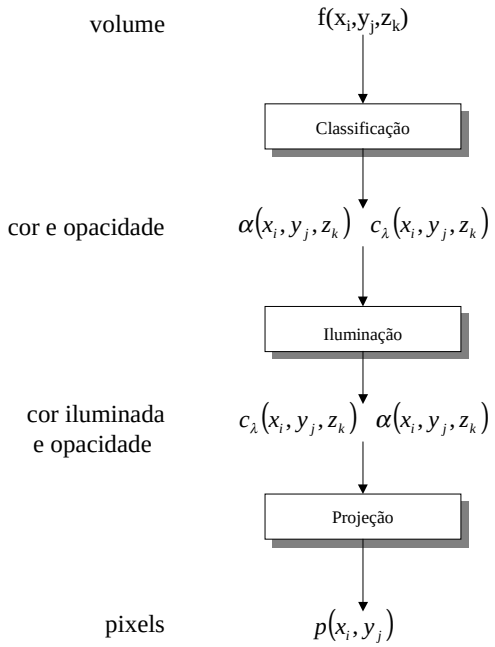


Figura 5.1 *Pipeline* de *rendering* de volumes

Nesta seção assumimos que os dados estão representados como valores escalares para pontos de um *grid* retilíneo. Através de considerações de interpolação desses valores discretos, podemos considerar que temos uma função escalar *f(x)* definida para todos os pontos (x_i, y_j, z_k) no volume. Assim, podemos verificar que o objetivo do processo de visualização direta de volumes é a criação de uma imagem $p(x_i, y_j)$ que represente uma

projeção da função $f(x_i, y_j, z_k)$. Este objetivo é atingido através de três etapas básicas, representadas no diagrama da Figura 5.1.

A primeira etapa do processo de *rendering* de volumes é a classificação dos dados. Ela permite ao usuário encontrar as estruturas existentes no conjunto de dados, podendo isolá-las e definir sua forma e extensão. A classificação envolve o mapeamento dos valores escalares a serem visualizados, em valores de cor associados e valores de opacidade, o que é feito através de funções de mapeamento que são específicas para cada aplicação (funções de transferência).

A opacidade é uma medida da transparência de cada *voxel*, descrevendo a quantidade de luz incidente no *voxel* que é absorvida por ele. Em geral esta propriedade varia entre 0 e 1, sendo o valor 1 associado a *voxels* que absorvem toda a luz incidente (completamente opacos).

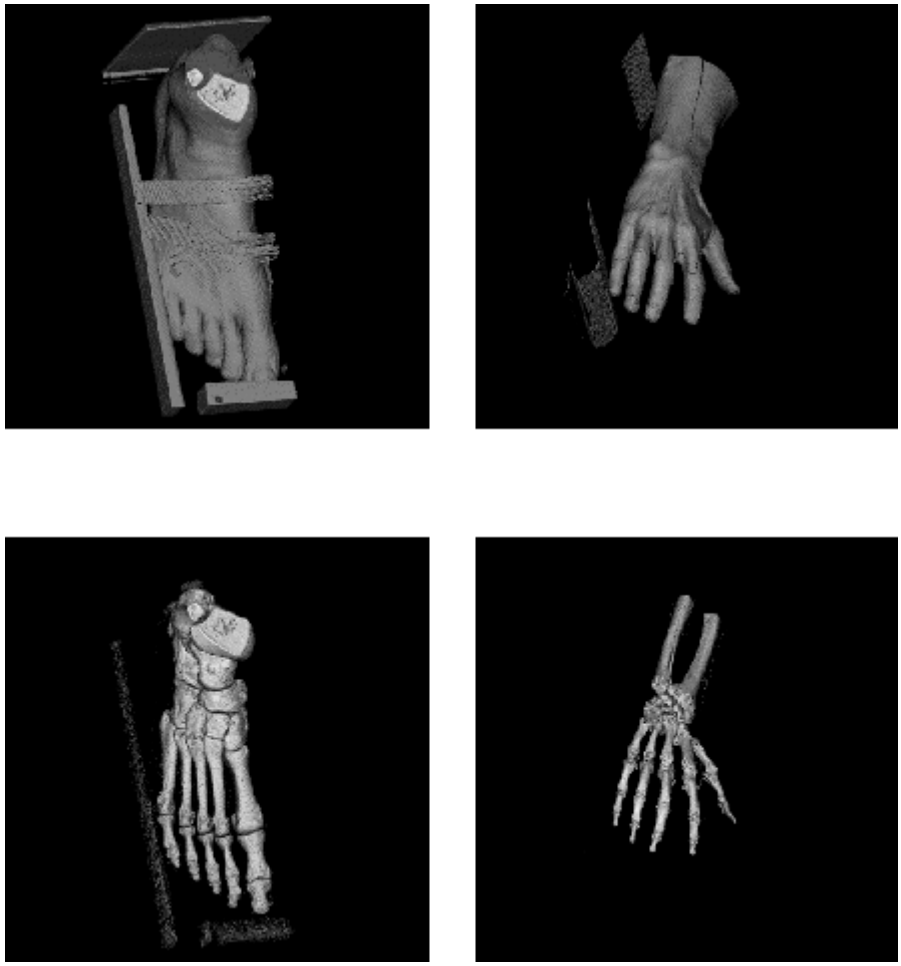


Figura 5.2 Volumes com diferentes funções de transferência

O mapeamento do valor escalar em cor é feito para gerar uma representação visual do material que representa cada *voxel*. Já o mapeamento em valores de opacidade é usado para possibilitar a exibição ao observador apenas da parte do volume que lhe interessa, tornando transparentes as porções que não precisarem ser visualizadas. Assim, permite-se a visualização de estruturas internas ao volume. Resumindo, a função do processo de classificação é possibilitar a identificação do material que compõe cada *voxel.* A Figura 5.2 representa a visualização do mesmo volume de dados com diferentes funções de transferência para a opacidade.

O cálculo da iluminação é a etapa do processo de *rendering* que tem o objetivo de descrever a aparência dos objetos. Este processo descreve a forma como se dá a interação dos diferentes tipos de luz, de diferentes fontes luminosas, em diferentes posições, com os materiais dos *voxels* que compõem o volume. Esta etapa modifica a cor especificada para cada *voxel* pelo mapeamento da função de transferência, de modo a realçar a percepção tridimensional na imagem final. Em geral são usadas adaptações dos modelos de iluminação utilizados em computação gráfica.

A terceira etapa consiste em visualizar um volume de valores escalares representando a cor de cada *voxel*, já considerando a iluminação. Nesta etapa verifica-se como cada *voxel* contribui para o valor dos *pixels*. Este processo envolve a utilização dos métodos de reconstrução volumétrica (interpolação) e uma modelagem da forma como a imagem do volume será formada.

Nas seções seguintes detalharemos cada uma das etapas deste processo de visualização, usando para isto uma abordagem semelhante à utilizada pelo algoritmo *Ray Casting*, de modo que, partindo da etapa de projeção, chegaremos ao processo de classificação.

5.1 Projeção

Esta etapa do processo de visualização recebe um volume (vetor 3D) de valores escalares (valores de cor e opacidade em cada ponto) e gera uma imagem resultante da projeção e acumulação das contribuições dos valores de cada um dos *voxels*. Para isto, modela-se este processo como uma aproximação do cálculo da propagação de luz através de um meio participativo (volume). O meio pode ser modelado como um bloco de um gel

semi-transparente com os valores de cor e opacidade representados a partir das amostras do volume classificado. Ao percorrer este meio, a luz interage com o gel, podendo ser absorvida, espalhada ou emitida pelo meio (Figura 5.3).

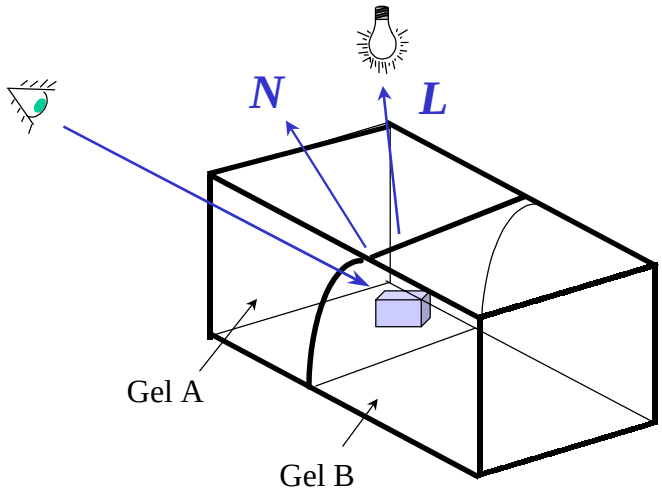


Figura 5.3 Modelo de iluminação no *voxel*

Se considerarmos somente os processos de interação luz/meio, podemos trabalhar com a aproximação da ótica geométrica. Assim, a interação da luz com os elementos de volume pode ser completamente descrita no âmbito da teoria de transporte (Krüger,1991). Isto significa que, para gerar a imagem, devemos integrar continuamente os efeitos das propriedades óticas ao longo de cada raio de visão, como mostra a Figura 5.4, que ilustra a modelagem física deste processo. O valor do *pixel (i,j)* é gerado pela integração dos efeitos de interação da luz com os *voxels* ao longo do raio de visão *r*.

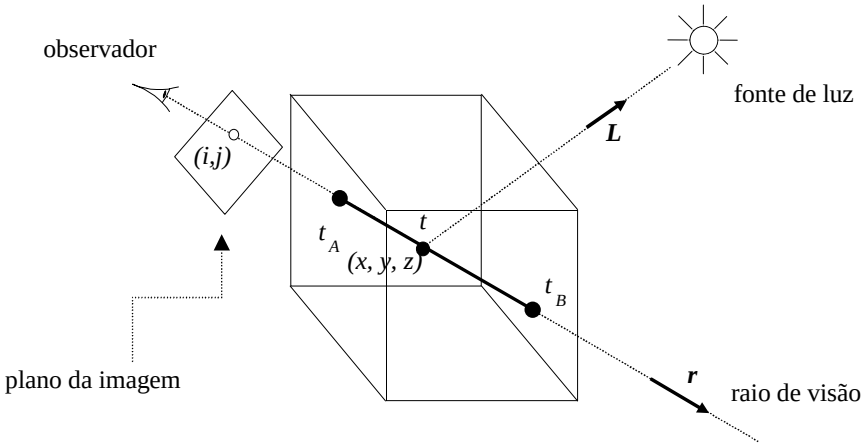


Figura 5.4 Modelo físico para o r*endering* de volumes

Nesta seção estudaremos como a imagem de um volume pode ser gerada. Para isto nos limitaremos ao estudo do modelo de câmera utilizado na computação gráfica. Ou seja, tentaremos reconstruir a imagem do volume como ele seria registrado por uma câmera fotográfica. Em seguida apresentaremos como a equação de integração dos efeitos de interação da luz com os *voxels* é deduzida, além de algumas técnicas de solução desta equação que levam a diferentes algoritmos de *rendering*.

5.1.1 Equação de Rendering de Volume

A equação que define a cor de um *pixel (i,j)* pode ser derivada integrando-se as contribuições de cor de cada um dos volumes elementares que são interceptados pelo raio ***r*** ilustrado na Figura 5.4.

Considerando-se a intensidade luminosa num certo comprimento de onda (*R*, *G* ou *B*) do volume por unidade de raio como sendo I_v (a ser estimada a seguir), temos que a intensidade da cor do *pixel (i,j)* neste comprimento de onda é obtido de:

$$I_{i,j} = \int_{t_A}^{t_B} e^{-\int_{t_A}^{t} \tau(s)ds} \; I_v(t)dt$$

Para a obtenção desta equação são admitidas as seguintes hipóteses:

- todos os fótons que atingem a imagem foram refletidos apenas uma vez, no objeto visível;
- não é considerada a atenuação da luz pelos *voxels* que ficam entre a fonte luminosa e o *voxel* sendo iluminado;
- é considerada a absorção isotrópica.

Esta equação pode ser derivada supondo-se que a luz emitida I_v seja atenuada até atingir o plano de projeção de acordo com o seguinte modelo de absorção. Considere que a energia luminosa I_v esteja atravessando um volume cilíndrico de área A_t e altura Δs onde existem n partículas de área A_p que impedem a passagem da luz, conforme ilustra a Figura 5.5.

Supondo-se que a espessura Δs seja suficientemente pequena e que as partículas não obscureçam umas às outras, a redução de energia luminosa pode então ser calculada por:

$$\Delta I_v = -\frac{nA_p}{A_t} I_v = -\frac{(\rho(s)A_t\Delta s)A_p}{A_t} I_v = -\rho(s)A_p I_v \Delta s = -\tau(s) I_v \Delta s\,,$$

sendo $\rho(s)$ a densidade de partículas opacas e $\tau(s) = \rho(s)A_p$ o coeficiente de oclusão por unidade de comprimento ao longo do raio.

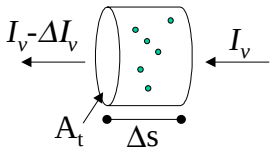


Figura 5.5 Modelo de absorção

No limite em que Δs tende para zero, temos:

$$\frac{dI_v}{ds} = -\tau(s) I_v \quad ,$$

que integrada resulta em:

$$I_t = I_v(t) e^{-\int_{t_A}^{t} \tau(s) ds}$$

Integrando todas as contribuições I_t ao longo do raio $\boldsymbol{r}$ de t_1 até t_2 obtemos a equação de *rendering*.

A integral da equação de *rendering* de volumes pode ser avaliada usando-se a soma de Riemann sobre uma partição $t_0 = t_A < t_1 < t_2 < \ldots < t_n = t_B$, de onde obtemos:

$$I_{i,j} \cong \sum_{k=0}^{n-1} e^{-\sum_{l=0}^{k-1} \tau_l \Delta t_l} I_k \Delta t_k = \sum_{k=0}^{n-1} \left(I_k \Delta t_k \cdot \prod_{l=0}^{k-1} e^{-\tau_l \Delta t_l} \right),$$

com:

$$I_k \equiv I\left(\frac{t_k + t_{k+1}}{2}\right)$$
$$\tau_l \equiv \tau\left(\frac{t_l + t_{l+1}}{2}\right)$$

Definindo: $\alpha_l \equiv 1 - e^{-\tau_l}$, **opacidade** da amostra *l;* $C_k \equiv \left(I_k \big/ \alpha_k \right) \Delta t_k$, **cor** da amostra *k* e $c_k \equiv C_k \alpha_k$ a multiplicação dos dois, obtemos:

$$I_{i,j} \cong \sum_{k=0}^{n-1}\left(c_k \cdot \prod_{l=0}^{k-1}(1-\alpha_k) \right)$$

Esta equação pode ser avaliada de duas formas distintas. No seguinte fragmento de código o somatório é feito de trás para a frente:

```
intensidade = c0
for( k = 0;  k < n; k =  k + 1 )
  intensidade  =  intensidade * (1-αk)  +  ck
```

Algoritmo 5-1 Composição de trás para a frente (*backward*).

Alternativamente, podemos escrever o somatório da frente para trás. Neste caso necessita-se de uma variável adicional para armazenar a transparência acumulada. Este procedimento tem a vantagem de poder terminar o somatório se a transparência se tornar muito pequena, de modo a não permitir que a luz proveniente de *voxels* distantes atinjam o olho do observador. O código correspondente é:

```
intensidade  =  cn
for(k = n; k > 0 e transparência > ε; k =  k - 1)
  intensidade = intensidade +  ck *  transparência
  transparência  =  ck * transparência
```

Algoritmo 5-2 Composição da frente para trás (*forward*).

Esta forma de cálculo da equação de *rendering* pode ser escrita com a utilização do operador *over* de composição digital (Porter,1984), o que resulta em:

$$\begin{aligned} I_{i,j} &= c_0 + c_1(1-\alpha_0) + c_2(1-\alpha_0)(1-\alpha_1) + \cdots + c_{n-1}(1-\alpha_0)\ldots(1-\alpha_{n-2}) \\ &= c_0 \ over\ c_1 \ over\ c_2 \ over\ c_3 \cdots over\ c_{n-1} \end{aligned}$$

Outra opção útil em implementações paralelas é a composição em árvore, que pode ser representada pela equação:

$$I_{i,j} = (c_0 \ over\ c_1)\ over\ (c_2 \ over\ c_3) \cdots over\ (c_{n-2} \ over\ c_{n-1})$$

5.2 Iluminação

O processo de iluminação é utilizado para melhorar a aparência 3D da imagem através da criação de uma ilusão de profundidade. Esta ilusão de profundidade é obtida

através da variação da cor dos objetos de acordo com a sua distância em relação ao plano da imagem, o material constituinte e a orientação com respeito à fonte de luz e ao observador. A forma de efetuar estes cálculos da cor dos objetos é definida de acordo com um modelo, denominado modelo de iluminação.

Os modelos de iluminação para computação gráfica são divididos em duas classes: locais e globais. Os modelos globais geram imagens de melhor qualidade, pagando o preço de serem computacionalmente mais caros. Em geral estes modelos tentam levar em consideração a luz refratada, que é transmitida através dos objetos, e as componentes especular e difusa da luz refletida. Alguns trabalhos propõem a utilização de modelos globais de iluminação para o *rendering* de volumes em aplicações como sistemas de visualização científica e cenas sintetizadas. No entanto, a maioria dos sistemas de visualização utiliza os métodos de iluminação local descritos a seguir.

Os modelos locais de iluminação baseiam-se apenas na reflexão da luz na superfície do objeto. A Figura 5.3 apresenta as componentes de iluminação na interação da luz com a superfície do objeto. Nesta figura fica claro que é fundamental determinar a orientação da superfície do objeto a ser visualizado.

Os modelos locais de iluminação somente computam a reflexão difusa e especular da luz emitida pelas fontes luminosas sobre a superfície do objeto.

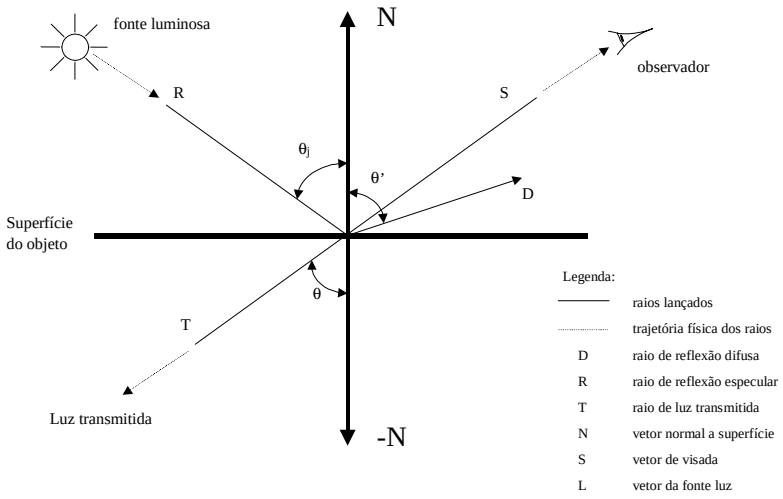


Figura 5.6 Relações da luz com a superfície do objeto.

Um modelo bastante popular foi proposto por Phong (Phong,1975). Nele, a reflexão difusa é causada pela absorção e reirradiação uniformemente distribuída da luz a partir da superfície iluminada. O efeito é modelado pela lei de Lambert, descrita pela equação:

$$I = I_l k_d (N \bullet L),$$

onde I é a intensidade refletida, I_l é a luz incidente, k_d é o coeficiente de reflexão difusa característico do material e N é o vetor normal à superfície no ponto em questão. Para situações práticas, a fonte de luz é modelada como um termo constante (luz ambiente) e um termo linear; assim, o modelo pode ser descrito por:

$$I = I_a k_a + I_l k_d (N \bullet L),$$

onde I_a é a intensidade da fonte de luz e k_a é a constante de reflexão ambiental. Se considerarmos ainda a componente especular, temos o modelo de iluminação expresso pela equação:

$$I = I_a k_a + \frac{I_l (k_d (N \bullet L) + k_s (E \bullet R)^n)}{D + K},$$

onde d é a distância entre o observador e o objeto, K é uma constante arbitrária, R é o vetor da fonte de luz, E é o vetor do observador, k_s é o coeficiente de reflexão especular e n é o expoente de reflexão especular. Esta equação pode ser aproximada por:

$$I = I_a k_a + \frac{I_l \left(k_d (L \cdot N) + k_s (N \cdot H)^n\right)}{D},$$

onde H é o vetor normalizado que dá a direção de realce máximo (Blinn,1977), sendo definido como a metade do ângulo entre a fonte e o observador.

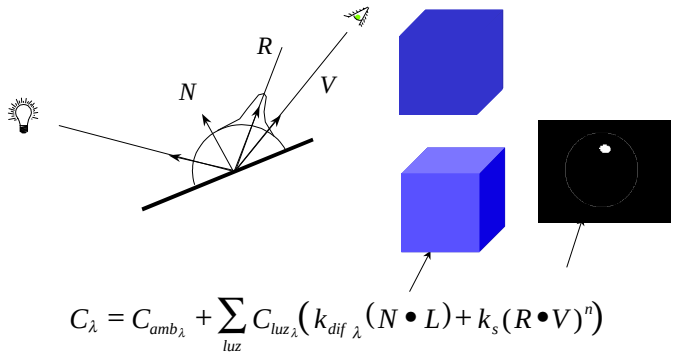


Figura 5.7 Modelo de iluminação de Phong

Este modelo de iluminação aplicado a um *voxel* pode ser melhor visualizado pela Figura 5.3, onde temos representadas a normal *N*, a direção da fonte de luz *L* e a direção do raio *r*.

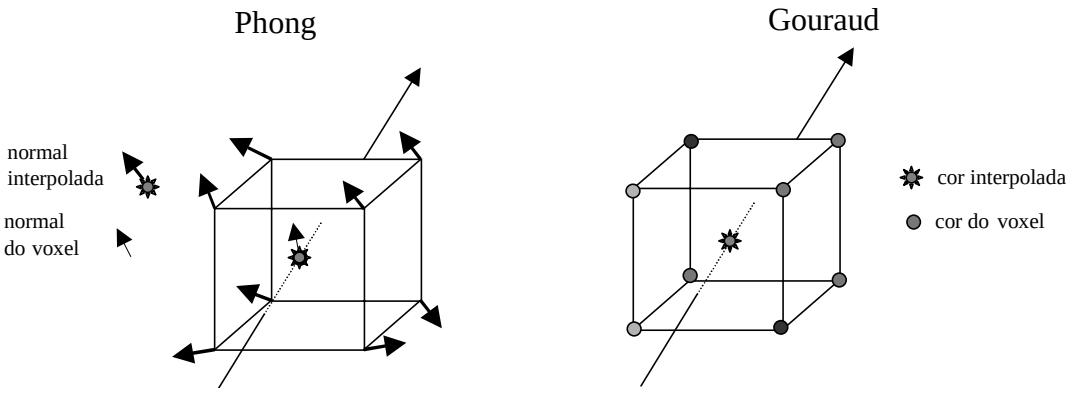


Figura 5.8 Algoritmos de Gouraud e Phong

Definido o modelo de iluminação, podemos utilizar um dos dois algoritmos de iluminação: Gourad (Gouraud,1971) ou (Phong,1975). O algoritmo de Gourad fornece a intensidade de luz no centro de cada *voxel* do volume, então usa esse valor para interpolar a iluminação em pontos intermediários. O algoritmo de Phong interpola o valor da normal e calcula a iluminação em cada ponto intermediário que surja. Esses modelos são utilizados para calcular a cor de cada *voxel* do volume. Isto está representado na Figura 5.8.

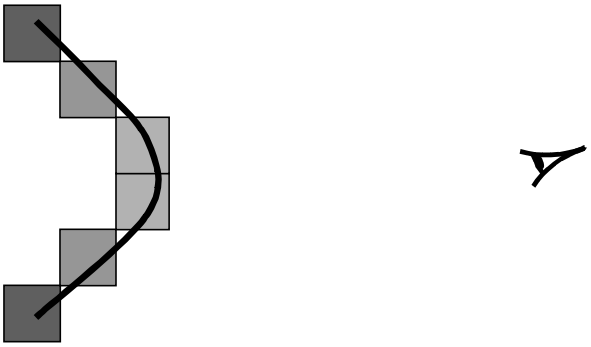


Figura 5.9 Algoritmo baseado na profundidade (*depth only*)

Uma alternativa é utilizar um algoritmo mais simples como o baseado na distância ao observador (*depth only*). Este não calcula a normal à superfície, apenas associa um valor de iluminação baseado em um critério de distância. Esta simplicidade é obtida com

o custo de obscurecer alguns detalhes do volume visualizado. A Figura 5.9 apresenta a idéia deste algoritmo. Trata-se de um algoritmo rápido, mas que realiza uma simulação muito primitiva da iluminação.

5.2.1.1 Estimativa da Normal

Um ponto central à aplicação dos algoritmos de iluminação é a necessidade de utilização da orientação da superfície. Isto é feito através da estimativa do vetor normal à superfície no ponto que será iluminado. Quando se trata de *rendering* de volumes não faz muito sentido falar em superfícies. No entanto, considera-se no cálculo da iluminação que o ponto a ser iluminado pertence a uma isosuperficie existente no volume e que passa pelo *voxel*. A Figura 5.10 mostra uma representação desta questão.

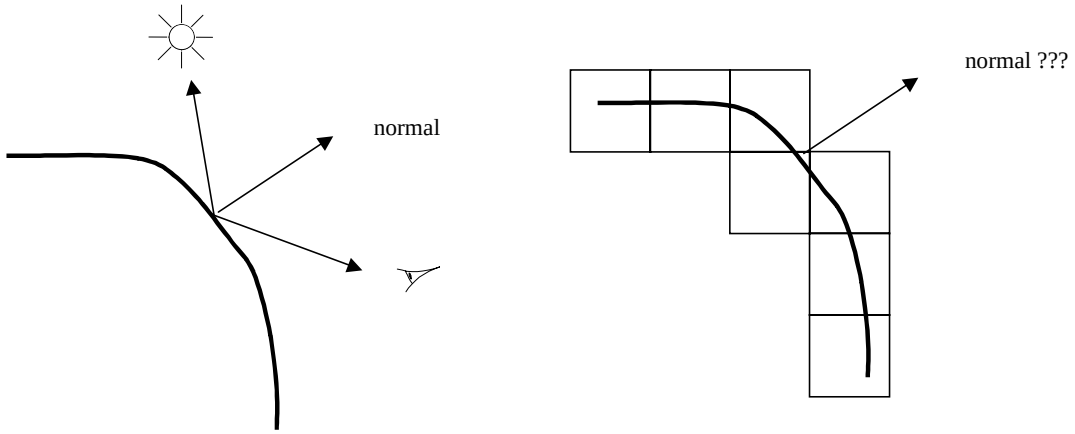


Figura 5.10 Estimativa da normal para iluminação

O método do gradiente ((Chen,1989), (Gordon,1983), (Gordon,1985) e (Reynolds,1985)) é um método de estimativa da normal à superfície que se processa no espaço da imagem. Ele gera uma imagem realista a um custo não muito alto. A chave do método é o uso do gradiente ao longo da direção *z*, que aproxima a variação na dimensão *z* entre vizinhos localizados no *z-buffer*. Após a determinação deste gradiente podemos aproximar a normal à superfície e, utilizando-a, calcular a intensidade de luz na superfície do objeto. O algoritmo estima a normal à superfície calculando o vetor gradiente $\nabla z = (\partial z/\partial u, \partial z/\partial v, 1)$. As derivadas $\partial z/\partial u$ e $\partial z/\partial v$ podem ser estimadas usando a diferença central, para a frente (*forward*), para trás (*backward*) ou ponderada (*weighted*).

Considerando uma posição *(i,j)* do *z-buffer*, temos:

- diferença para a frente: $\partial z / \partial u = z_i^{'} - z_{i+1}^{'}$;
- diferença para trás: $\partial z / \partial u = z_i^{'} - z_{i-1}^{'}$;
- diferença central: $\partial z / \partial u = z_{i+1}^{'} - z_{i-1}^{'}$

e a normal à superfície, **N**, é calculada usando $\nabla z \bullet N = 0$

Outra alternativa para a estimativa do gradiente baseia-se nos valores escalares do volume. Considere o dado volumétrico, representado por *V*, um campo escalar diferenciável, então temos:

$$V(x, y, z) = V(X),$$

usando-se o operador gradiente ∇, teremos:

$$\nabla V(X) = \left(\frac{\partial V}{\partial x}, \frac{\partial V}{\partial y}, \frac{\partial V}{\partial z} \right)$$

Para *V* definido em uma grade regular, o gradiente pode ser aproximado através de diferenças finitas por:

$$\left(\frac{1}{2}\left(V(x_{i+1}, y, z) - V(x_{i-1}, y, z)\right), \frac{1}{2}\left(V(x, y_{j+1}, z) - V(x, y_{j-1}, z)\right), \frac{1}{2}\left(V(x, y, z_{k+1}) - V(x, y, z_{k-1})\right) \right)$$

Desta forma, a determinação da normal *N* é feita com base na densidade *D*. Uma "superfície" é detectada pela avaliação da normal usando o gradiente do volume. As componentes desta normal são:

$$N_x = D(x+1, y, z) - D(x-1, y, z);$$

$$N_y = D(x, y+1, z) - D(x, y-1, z);$$

$$N_z = D(x, y, z+1) - D(x, y, z-1).$$

Este método é muito sensível à existência de ruídos, o que faz com que algumas vezes a aproximação não possa ser utilizada. Por isso, o gradiente pode ser calculado usando os seis *voxels* vizinhos ou todos os vinte e seis *voxels* (segunda ordem) da vizinhança considerada, conforme ilustrado na Figura 5.11.

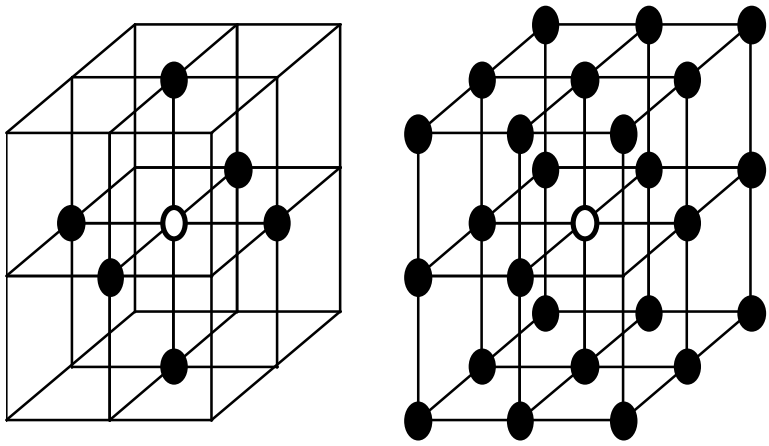


Figura 5.11 Vizinhança no cálculo do gradiente

A iluminação com o modelo de gradiente em tons de cinza, descrita em (Hohne,1986) também produz imagens de alta qualidade. A idéia básica é a utilização do gradiente em tons de cinza entre valores de *voxels* vizinhos no *z-buffer* para aproximar a normal à superfície. Esta abordagem utiliza a razão entre materiais que formam um *voxel* (i.e. o valor tonal do *voxel*) juntamente com as razões dos *voxels* vizinhos para calcular a profundidade e a direção da superfície que cruza o *voxel* em questão. O problema é que as normais variam em função da função de transferência.

Pommert *et al.* (Pommert,1990) propõem um algoritmo que se adapta à espessura do objeto contido no volume. Essencialmente, o algoritmo usa um critério simples baseado na freqüência local para escolher entre dois filtros de tamanhos diferentes. Estes introduzem a noção de escala; um filtro de segunda ordem calcula o gradiente em uma escala maior. Yagel *et al.* (Yagel,1992) realizam uma análise de descontinuidade dos vizinhos para escolher o operador de gradiente apropriado. Estratégias de decisão simples podem melhorar a precisão da estimativa da normal, segundo esse trabalho.

A estimativa da normal representa um grande esforço computacional, sendo algumas vezes realizada em um passo de pré-processamento.

5.3 Classificação dos Dados

Para os algoritmos de visualização volumétrica, podemos dizer que o passo de classificação é a etapa inicial. Isto se dá porque a manipulação do volume de dados é

independente da técnica que será empregada para a visualização, sendo utilizada como uma etapa de pré-processamento para a adequação do volume de dados.

Em algoritmos de *rendering* direto de volumes, a classificação é feita através de um mapeamento dos valores dos *voxels* em valores de cor e opacidade. Esta é uma tarefa sujeita a erros, e exige que o usuário possua um certo conhecimento acerca do que espera encontrar no volume. Em aplicações como imagens médicas, os dados e valores de componentes são razoavelmente conhecidos, logo este não é um problema crítico. No entanto, para aplicações como interpretação sísmica, esta etapa é crítica, pois *a priori* o analista não sabe o que encontrará no volume.

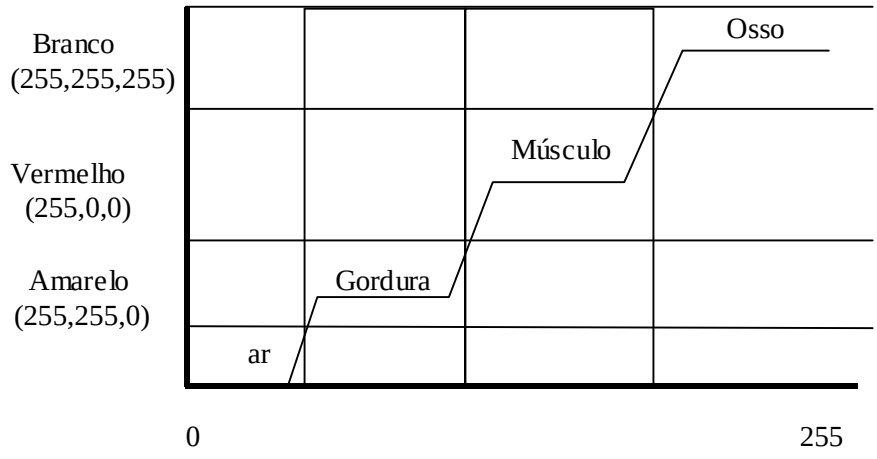


Figura 5.12 Mapeamento de cor dos materiais

Uma ferramenta eficiente para auxiliar o projeto da função de mapeamento é o histograma do volume. O histograma representa o número de vezes que ocorrem *voxels* de cada um dos valores possíveis (valores discretos de *voxels*). Assim, pode-se conhecer um pouco da estrutura dos dados do volume e decidir os valores que representam a estrutura que se deseja visualizar.

O modelo natural para realizar esse mapeamento consiste em considerar apenas a intensidade do *voxel*.

Um sistema típico baseado neste modelo de classificação foi proposto em (Drebin,1988) para o caso particular de dados de CT (raios X). Este esquema emprega um classificador probabilístico, baseado em semelhanças. Isto é realizado através da porcentagem de existência de um determinado material em cada *voxel*, a partir do que pode ser encontrado o valor da cor. As fronteiras entre partes internas do volume podem

ser detectadas pela variação de densidade (valor do *voxel*). Para obter a função de mapeamento utiliza-se um método probabilístico levando em consideração o número de materiais presentes no volume e a porcentagem de material em cada *voxel*. Neste esquema as funções de mapeamento dependem exclusivamente do valor de intensidade do *voxel*. A Figura 5.12 ilustra uma classificação de materiais típica e o mapeamento dessas densidades em cores.

A Tabela a seguir mostra um exemplo do mapeamento de densidade em materiais e destes em cores e opacidade.

Valor:	de 30 a 90	de 190 a 230	de 230 a 255
Material:	Gordura	Tecido	Osso
Cor (R,G,B):	255, 200, 20	255, 80, 60	255, 255, 255
Opacidade:	20 %	80 %	100 %

Tabela 5-1 Classificação dos materiais

Outra técnica de classificação foi proposta por Levoy (Levoy,1988). Este esquema de mapeamento inicia-se pela associação de uma opacidade α_v aos *voxels* que possuem valor f_v, associando opacidade zero aos outros *voxels*. Levoy utiliza uma queda gradual da opacidade à medida que a função de transferência se afasta do valor especificado na razão inversa da magnitude do vetor gradiente local. O mapeamento é conseguido com a utilização da expressão:

$$\alpha(x_l) = \alpha_v \begin{cases} 1 & se\ |\nabla f(v_l)| = 0\ e\ f(v_l) = f_v \\ 1 - \frac{1}{r}\left|\frac{f_v - f(v_l)}{|\nabla f(v_l)|}\right| & se\ |\nabla f(v_l)| > 0\ e\ f(v_l) - r|\nabla f(v_l)| \le f_v \le f(v_l) + r|\nabla f(v_l)| \\ 0 & de\ outra\ forma \end{cases}$$

A Figura 5.13 ilustra a forma de uma função de mapeamento gerada com este esquema.

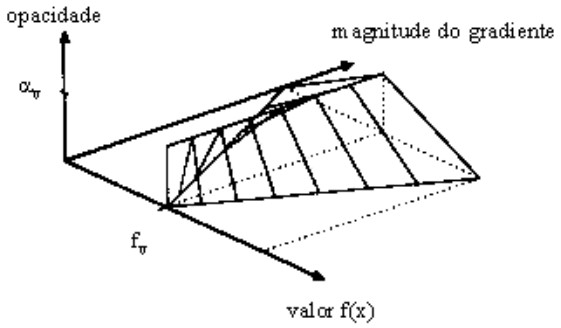


Figura 5.13 Função de mapeamento gerada pelo esquema de Levoy

Upson e Keeler (Upson,1988) definiram duas funções de transferência, uma para cor e outra para opacidade, para a faixa de valores escalares do volume (Figura 5.14). Esses valores eram utilizados junto com os gradientes obtidos para o cálculo de luz e sombra na imagem final.

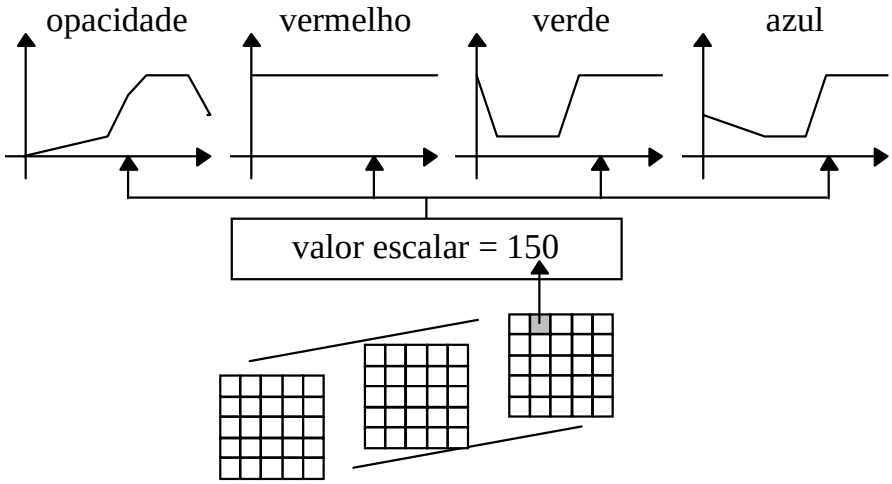


Figura 5.14 Mapeamento da densidade em RGB e opacidade

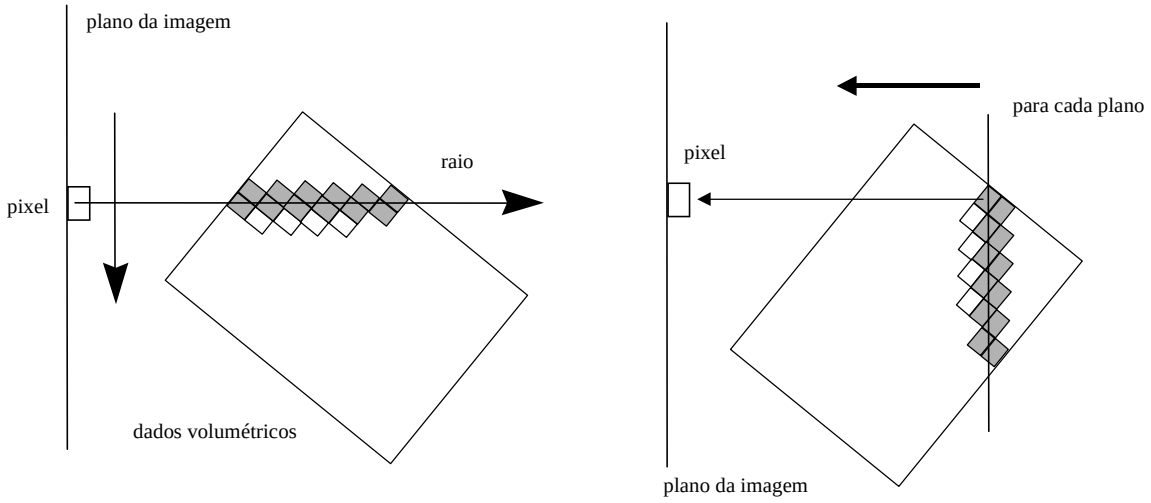
Um método de classificação mais sofisticado foi apresentado por (He,1996). Em seu trabalho são discutidos métodos estocásticos para a geração das funções de transferência. A abordagem define o que deve ser visualizado através de medidas objetivas, como entropia máxima da imagem ou variação do histograma. Alguns resultados mais avançados são descritos em (Marks,1990).

Apesar desses esforços, podemos verificar que não existe uma regra geral para o projeto das funções de transferência. Sendo assim, é importante que sejam fornecidas ferramentas ao usuário para realizar este projeto interativamente, de preferência sobre o histograma do volume. Dessa forma, por tentativa e erro, o usuário pode convergir para a visualização desejada.

6. Algoritmos de Rendering de Volume

Existem quatro abordagens básicas para o problema de *rendering* direto de volumes: ordem da imagem (*forward mapping*), ordem dos objetos (*backward mapping*), métodos baseados em mudança de base e métodos baseados em transformações geométricas do volume.

Na primeira abordagem geralmente lança-se um raio associado a cada *pixel* da imagem e encontram-se os *voxels* interceptados, os quais contribuem para o valor do *pixel* (Figura 6.1a). Já a abordagem baseada na ordem dos objetos percorre o volume e, para cada *voxel*, encontra os *pixels* que são afetados pela sua contribuição (Figura 6.1b).



(a) Espaço da imagem (b) Espaço do objeto

Figura 6.1 Algoritmo do espaço da imagem e do objeto

Os métodos baseados em transformações geométricas do volume aplicam transformações geométricas afins ao volume, de modo a gerar uma simplificação das situações anteriores, com o objetivo de otimizar o processo. De maneira semelhante, os métodos baseados em mudança de base realizam uma transformação do volume para outro sistema de coordenadas e, então, empregam uma das duas estratégias iniciais para percorrer o volume e gerar a imagem.

O algoritmo de *ray casting*, representante da primeira classe de algoritmos, baseia-se em raios lançados do ponto de vista do observador passando através de cada *pixel* da tela e interceptando o volume. Se o raio interceptar o volume, o conteúdo do

volume ao longo do raio é amostrado, transformando-se o valor em cor e opacidade e resultando num valor que é atribuído ao *pixel*.

Os algoritmos *v-buffer* (Upson,1988) e *splatting* (Westover,1990), que trabalham na ordem dos objetos, projetam os *voxels* no espaço da tela. Esses métodos necessitam que os *voxels* sejam ordenados do mais distante para o mais próximo (*back-to-front*) ou vice-versa (*front-to-back*).

Como representantes da classe de algoritmos baseados em transformações afins do volume, temos os algoritmos que usam o cisalhamento das fatias (*slice shearing*). Eles primeiro aplicam rotações ao volume na memória até que este esteja alinhado com o plano de visualização. Depois, a visualização é feita percorrendo o volume, com caminhamento simplificado gerado pela transformação.

A outra classe de algoritmos, mais recente, contém os algoritmos baseados na transformação do volume para outro espaço (e.g. freqüências) a partir de onde é realizada a visualização.

Neste capítulo descrevemos os principais algoritmos para *rendering* de volumes. Inicialmente detalhamos o algoritmo *Ray Casting*, que gera imagens de muito boa qualidade, demandando um considerável esforço de processamento. Em seguida discutimos o algoritmo *Splatting* (Westover,1990), ordem dos objetos, que gera imagens de alta qualidade, sendo muito utilizado em implementações paralelas de *rendering* de volumes. Também é apresentada uma descrição do algoritmo de *rendering* de volumes baseado em textura que aproveita o *hardware* desenvolvido para o mapeamento de texturas 3D, propiciando um bom desempenho para a visualização de volumes. Como representante da classe dos algoritmos baseados em transformações do volume é descrito o algoritmo *Shear-Warping* (Lacroute,1996). No final, uma visão geral dos métodos baseados em mudança de base é apresentada.

6.1 Ray Casting

O algoritmo da ordem da imagem *Ray Casting*, é o mais usado para a visualização de volumes quando necessita-se de imagens de alta qualidade (Elvins,1992). O algoritmo percorre todos os *pixels* da imagem determinando a cor e a opacidade para cada um. Um raio é disparado de cada *pixel* através do volume. As opacidades e cores encontradas ao longo do raio são acumuladas até se determinar a cor e a opacidade final do *pixel* (Figura 6.2). Várias otimizações, melhorias e métodos

híbridos são citados na literatura, principalmente em ((Tuy,1984), (Levoy,1988), (Levoy,1990a), (Upson,1988) e (Seixas,1994).

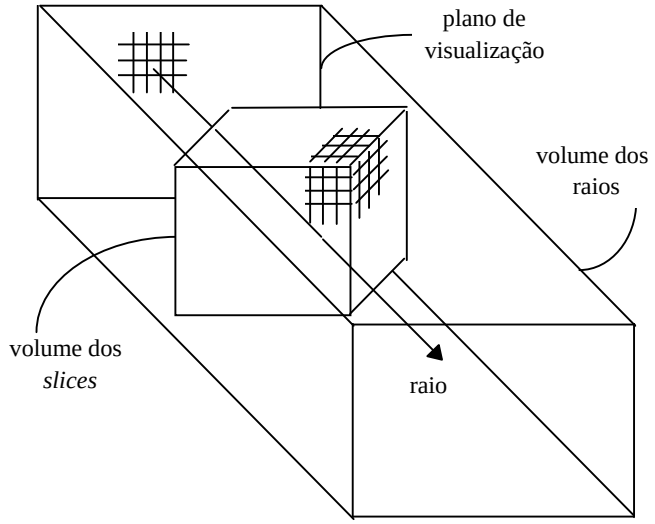


Figura 6.2 Lançamento do raio

Este algoritmo possui um alto custo computacional, mas pode ser paralelizado facilmente, uma vez que os valores dos *pixels* são determinados através do lançamento de raios independentes entre si.

O algoritmo pode ser definido em linhas gerais de maneira bastante simples, como mostra o Algoritmo 6.1.

```
InitRay( ) // inicializa  parâmetros de lançamento dos
raios
para  i de 1 a ImageHeight   // loop de lançamento
   para j de 1 a ImageWidth
       UpdateRay (ray, i, j )
       image[i,j] = RayCast(ray)
   fim para
fim para
```

Algoritmo 6.1 *Ray Casting*

A tarefa fundamental deste algoritmo é o lançamento dos raios, o qual pode ser feito de maneira eficiente se considerarmos o *frustum* de visão, lançarmos o primeiro raio como uma aresta desse *frustum* e, os demais, de maneira incremental. Para isto, consideremos o volume de visualização (*frustum*) para uma projeção paralela (Figura 6.3), definido por dois planos *near* e *far* e quatro pontos em cada plano definindo uma janela ((n_1, n_2, n_3, n_4) e (f_1, f_2, f_3, f_4)), todos descritos no sistema de coordenadas do objeto. Para gerar a imagem segundo o ponto de vista desejado aplicam-se as transformações necessárias aos pontos que definem o *frustum* e lançam-

se os raios a partir do plano da imagem rotacionado e escalado para a resolução especificada. Esta abordagem foi proposta em (Seixas,1997).

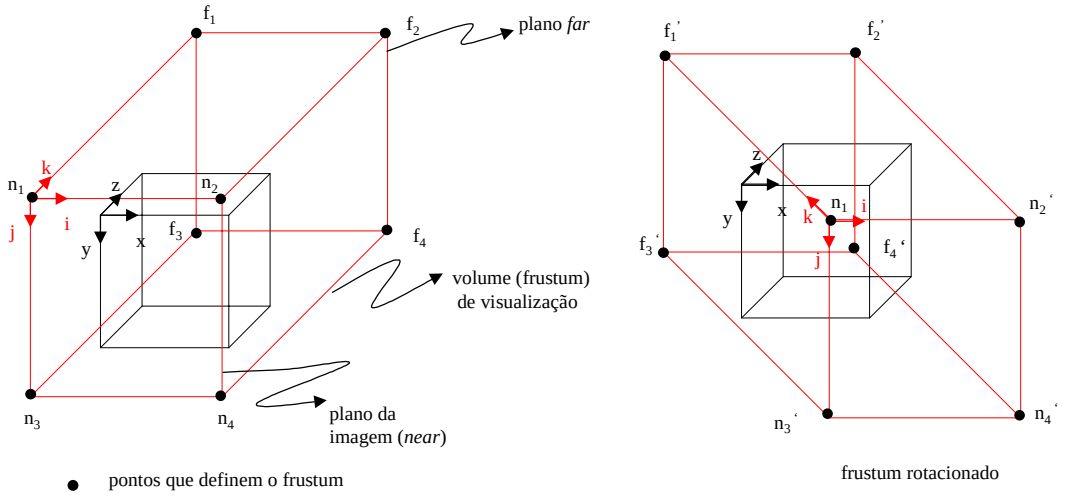


Figura 6.3 *Frustum* de visualização

Os raios são lançados de um plano para o outro em função dos seus pontos extremos. Assim, o primeiro raio liga o ponto n_1' ao ponto f_1'. Os demais raios são calculados através de incrementos obtidos para o deslocamento de um *pixel* na *scanline* (Δi) e para deslocamento de uma *scanline* para outra (Δj). Isto é calculado em função do tamanho do volume de visualização, em *pixels*, segundo a direção de lançamento do raio inicial, nos três eixos. Alguns cuidados na definição do tamanho da imagem devem ser tomados; em geral procura-se gerar imagens com dimensão máxima próxima do valor da maior diagonal do volume, de modo a evitar *aliasing*.

O Algoritmo 6.2 mostra como se dá esse esquema de lançamento, implementado nas funções substituindo as chamadas às funções *InitRay* e *UpdateRay* do Algoritmo 6.1 pelo código que implementa o lançamento incremental do raio.

Neste algoritmo os raios são calculados através de seus pontos de entrada e saída no volume de visualização *(frustum)*; assim, a função *raycast* é quem efetivamente realiza o lançamento do raio especificado ao longo do volume.

```
// calcula os incrementos
```

$$\Delta j_{comp} = \left(n3_{comp} - n1_{comp} \right) \Big/ Tela_{comp}, \quad comp = x, y, z \quad e \quad Tela_z = \min(Tela_x, Tela_y)$$

$$\Delta i_{comp} = \left(n2_{comp} - n1_{comp} \right) \Big/ Tela_{comp}, \quad comp = x, y, z \quad e \quad Tela_z = \min(Tela_x, Tela_y)$$

```
// definição do primeiro raio a ser lançado
```

$$\vec{p}_1 = \vec{n}_1 \qquad \vec{p}_2 = \vec{f}_1$$

```
  // lança os raios
  para i = 0 até ImageHeight     // loop de lançamento dos
raios
     para j = 0 até ImageWidth
          p1 = n1 + Δi * pixel + Δj * scanline
          p2 = f1 + Δi * pixel + Δj * scanline
        image[i, j] = raycast( p2,  p1 )
     fim para j
  fim para i
```

Algoritmo 6.2 Lançamento dos raios

No Algoritmo 6.1, os pontos $\vec{p}_1$ e $\vec{p}_2$ definem o raio e Δi e Δj representam o incremento nas coordenadas dos pontos que definem o raio, para um deslocamento unitário na imagem ao longo das direções *i* (*scanlines*) e *j* (*pixels*).

A função *raycast*, que compõe as contribuições dos *voxels* ao longo do raio, inicialmente verifica se o raio em questão realmente penetra no volume. Isto pode ser realizado através de um algoritmo de cálculo da interseção do raio com a *bounding box* do volume. Como a maioria dos raios intercepta o volume de dados, podemos utilizar o algoritmo de Cyrus-Beck (Foley,1990), que é bastante eficiente na determinação dos pontos de interseção.

O algoritmo de Cyrus-Beck baseia-se na determinação de um parâmetro *t* da interseção do raio com a superfície do volume de dados e na classificação deste ponto como potencialmente entrando (*PE*) ou potencialmente saindo (*PS*), de acordo com a superfície de interseção. O valor *t* é dado por:

$$t = \frac{-N_i \bullet [P_1 - P_i]}{N_i \bullet [P_2 - P_1]},$$

onde N_i é a normal da face, P_1 é o ponto inicial do raio, P_2 é o ponto final do raio e *Pi* é um ponto arbitrário a ser classificado como *PE* ou *PS*. Desta forma, se o denominador é maior que *0*, então tem-se o caso *PS* e, se o denominador é menor que *0*, temos o caso *PE*.

Os pontos de interseção com o volume ficam assim determinados por uma tabela de cálculo de interseções (Tabela 6-1).

Na Tabela 6-1, x_{min}, x_{max}, y_{min}, y_{max}, z_{min} e z_{max} representam as dimensões do volume dos dados. Como em geral consideramos o volume com mínimo na origem

temos $x_{\min} = y_{\min} = z_{\min} = 0$. Os pontos de partida dos raios no plano de visualização são (x_0, y_0, z_0), e Δx, Δy e Δz representam os tamanhos ao longo dos eixos respectivos.

Face	Ponto de interseção
Left	$t = \frac{x_0 - x_{\min}}{-\Delta x}$
Right	$t = \frac{x_{\max} - x_o}{\Delta x}$
Bottom	$t = \frac{y_0 - y_{\min}}{-\Delta y}$
Top	$t = \frac{y_{\max} - y_o}{\Delta y}$
Near	$t = \frac{z_0 - z_{\min}}{-\Delta z}$
Far	$t = \frac{z_{\max} - z_0}{\Delta z}$

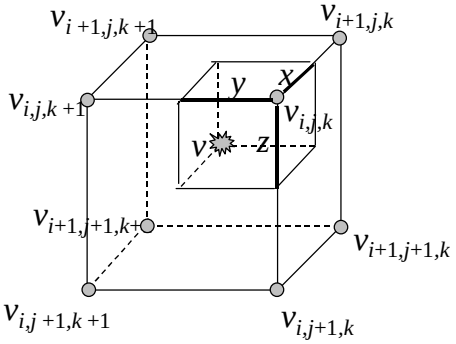
Tabela 6-1 Cálculo de Interseções

Finalmente os raios que interceptam o volume são realmente lançados. No lançamento verifica-se que os pontos de amostragem no raio não coincidem com as posições do volume que possuem valor conhecido (*voxels*). Existem várias soluções para tratar este problema; algumas envolvem a interpolação do valor do campo escalar do volume enquanto outras apenas realizam um caminhamento discreto ao longo do volume.

Em geral, percorre-se o raio em pontos de amostragem eqüidistantes, interpolando o valor do campo escalar a partir dos valores dos *voxels* vizinhos. Este procedimento realiza uma amostragem do raio com intervalos unitários e utiliza uma das técnicas de interpolação discutidas no capítulo anterior. A Figura 6.4 mostra como pode ser interpolado o valor no ponto *v* (interpolação trilinear).

Este procedimento necessita da determinação do *voxel* correspondente a cada ponto a ser amostrado. Para evitar este problema, pode ser utilizado o algoritmo de rasterização do raio no volume discreto, de modo a determinar de forma incremental quais *voxels* do volume são interceptados pelo raio. Desta forma, evitamos o problema de localização e garantimos que, para um mesmo raio, não existem duas amostragens

para um mesmo *voxel*. Essa abordagem permite duas variações para a escolha do campo escalar do volume representante da amostra do raio. Pode-se realizar a interpolação dos valores dos *voxels* vizinhos no centro de cada elemento de volume ou tomar um valor representativo para cada *voxel* e utilizá-lo sem interpolação.



$$v(x,y,z) = (1-x)(1-y)(1-z)v_{i,j,k} + (x)(1-y)(1-z)v_{i+1,j,k} + (1-x)(y)(1-z)v_{i,j+1,k} + (x)(y)(1-z)v_{i+1,j+1,k} + (1-x)(1-y)(z)v_{i,j,k+1} + (x)(1-y)(z)v_{i+1,j,k+1} + (1-x)(y)(z)v_{i,j+1,k+1} + (x)(y)(z)v_{i+1,j+1,k+1}$$

Figura 6.4 Exemplo de interpolação trilinear no volume

Uma alternativa para a rasterização do raio no volume discreto é a utilização do algoritmo de Bresenham (Foley,1990), originalmente desenvolvido para desenhar linhas em um dispositivo matricial.

Entretanto, o algoritmo de Bresenham não garante a contribuição de todos os *voxels* interceptados por um raio, conforme ilustrado na Figura 6.5. Dependendo do método de interpolação e do algoritmo de iluminação adotados, essa característica pode ser bastante prejudicial à visualização volumétrica, pois pode levar à visualização de partes que normalmente não seriam visualizadas ou até à visualização de artefatos.

Outra abordagem, proposta por Cohen (Cohen,1994), tem o objetivo de garantir a amostragem de todos os *voxels* interceptados pelo raio. Esta abordagem utiliza o conceito de *tripod*, segundo a qual cada *voxel* ao longo do raio é adjacente, através de uma de suas faces, ao seu predecessor. Ou seja, o *voxel* seguinte a ser amostrado pode ser determinado através da face perfurada pelo raio na saída do *voxel* atual.

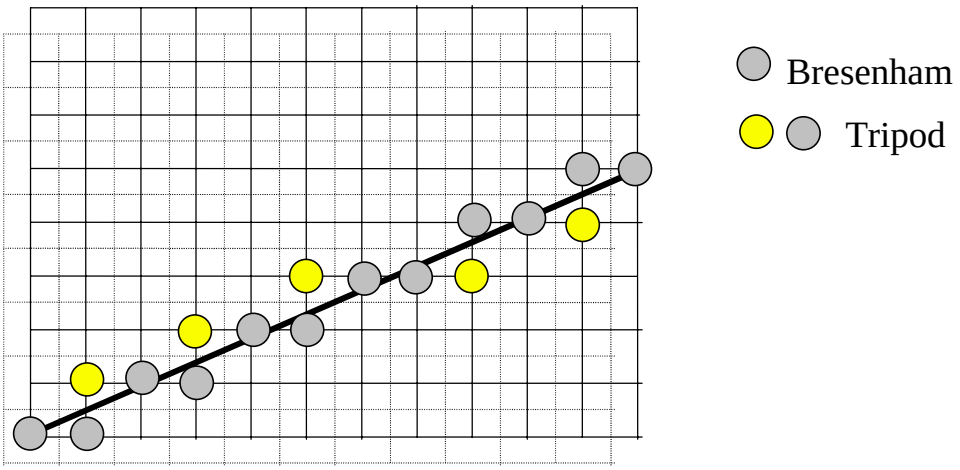


Figura 6.5 Algoritmo de Bresenham e tripod.

Admitindo que as coordenadas do raio crescem ao longo das três coordenadas, verificamos que o raio só pode sair do *voxel* por uma das três faces mostradas na Figura 6.6a. Essas três faces geram um conjunto de três arestas (*L1*, *L2* e *L3*) adjacentes pelo vértice *O*, Figura 6.6b. Usando a técnica do ponto médio (Aken,1985) e considerando a projeção do raio nos planos *xy*, *xz* e *yz* do *voxel*, avaliamos a equação do raio projetado no ponto *O*, gerando três valores e_{xy}, e_{yz} e e_{xz}. Através de testes desses valores podemos determinar qual a face de saída, conforme apresentado no Algoritmo 6.1.

```
/ / percorre o raio de (x,y,z)  a ( x + dx, y+dy,z+dz)
Raio ( x, y, z, dx, dy, dz)
início
    sx = sinal (dx);        sy = sinal (dy);        sz = sinal (dz);
    ax = |dx| ;             ay = |dy| ;             az = |dz| ;
    bx = 2*ax;              by =2*ay;               bz = 2*az;
    exy = ay - ax;          exz = az - ax;          ezy = ay – az;
    n  = ax + ay + az;
    Enquanto (n > 0)
       VisitaVoxel (x, y, z );
       se  (exy < 0) então
          se  (exz < 0) então
             x = x + sx ;       exy = exy + by;      exz = exz + bz;
          senão
             z = z + sz
             exz = exz - bx;    ezy = ezy + by;
          fim se
        senão
          se (ezy < 0 ) então
             z = z + sz ;       exz = exz - bx;      ezy = ezy + by;
          senão
             y = y + sy
             exy = exy - bx;    ezy = ezy - bz;
          fim se
```

```
        fim se
    fim enquanto
fim
```

Algoritmo 6.3 Tripod

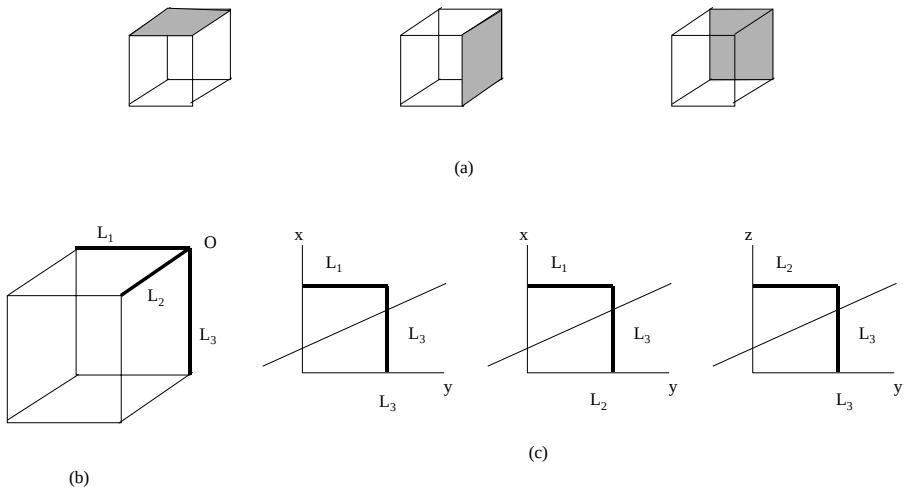


Figura 6.6 Conceito de tripod.

A face interceptada pode ser determinada pela relação entre as arestas que partilham o vértice comum e o raio, conforme ilustrado na Figura 6.6b.

A diferença entre as duas abordagens pode ser claramente verificada, comparando-se a Figura 6.7a, que utiliza o algoritmo de Bresenham, e a Figura 6.7b, que utiliza o conceito de *tripod*.

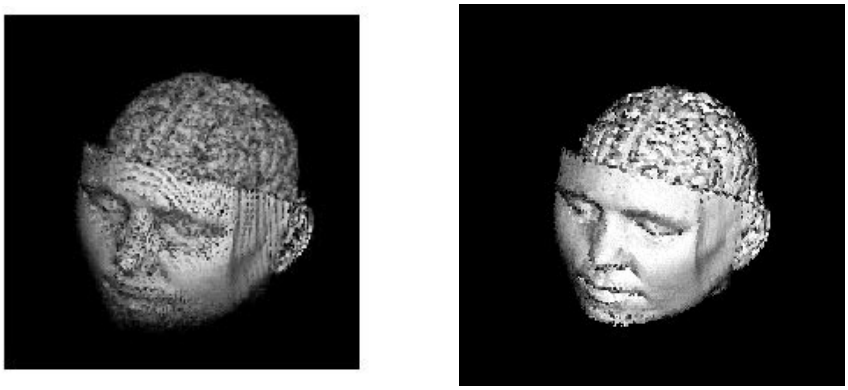


Figura 6.7 Imagem gerada com Bresenham e *tripod*

Uma vez definido o valor da amostra do volume ao longo do raio, podemos realizar as tarefas de classificação e cálculo da iluminação, conforme discutido no capítulo anterior. Uma forma alternativa de execução deste algoritmo lança os raios sobre volumes iluminados e classificados em

um passo de pré-processamento, de modo que o valor da amostra já determina a cor ou opacidade do volume (R,G,B,α) naquele ponto.

As amostras do raio são então compostas progressivamente através da utilização do operador de composição adequado, conforme discutido no capítulo anterior.

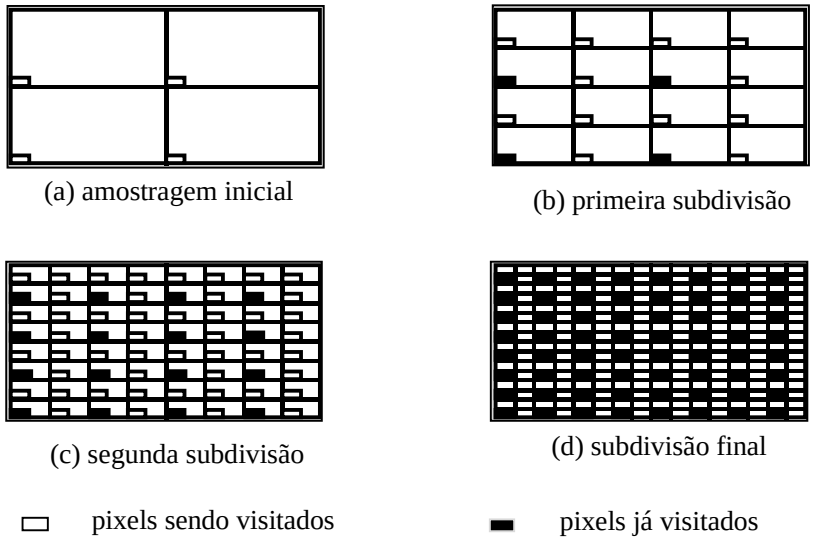


Figura 6.8 Refinamento progressivo

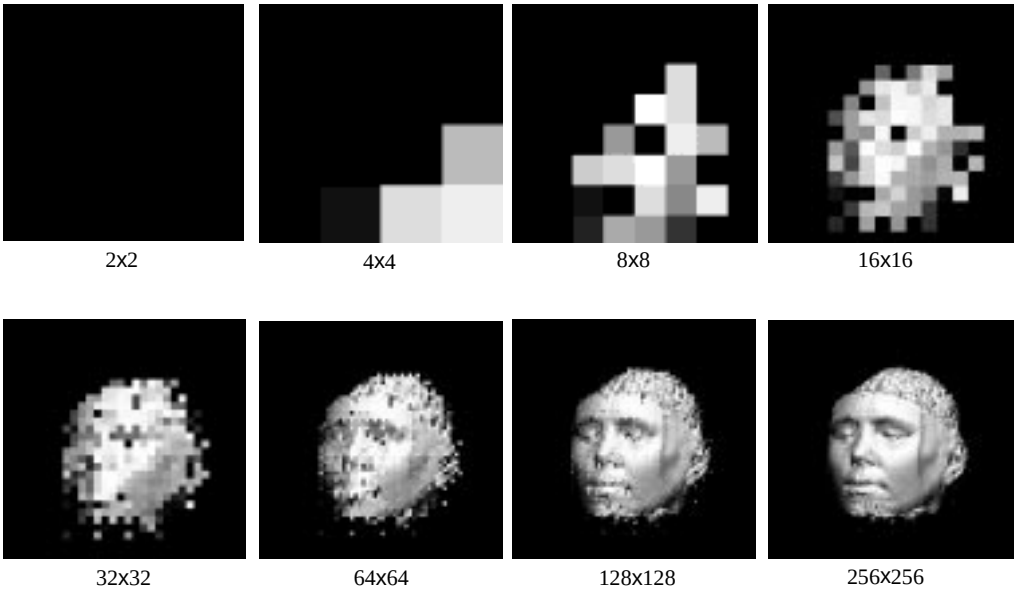


Figura 6.9 Exemplo de geração progressiva da imagem

Uma característica deste algoritmo é a facilidade de implementação da criação progressiva da imagem. A idéia é percorrer uma grade regular sobre o plano de visualização, diminuindo a cada etapa o passo na grade. Assim, a imagem é amostrada

rapidamente em baixa resolução (Figura 6.8a), sendo refinada até a resolução final (Figura 6.8d). Cada *pixel* da imagem é calculado uma única vez, ficando o tempo de cálculo da cor dos *pixels* inalterado. A Figura 6.9 apresenta um exemplo de geração de imagem com refinamento progressivo.

6.2 Splatting

O algoritmo *splatting* ((Westover,1990) e (Westover,1991)) é inspirado na estrutura do *pipeline* de *rendering* de polígonos, em que cada primitiva passa ao longo dos vários estágios por vez. Neste algoritmo, cada elemento é mapeado no plano da tela; em seguida, através de um processo de acumulação, tem sua contribuição adicionada à formação da imagem. O algoritmo termina quando todas as primitivas tiverem sido mapeadas na tela.

O algoritmo é definido por quatro etapas principais: transformação, classificação/iluminação, reconstrução e visibilidade.

O processo de transformação é composto pelo mapeamento das coordenadas *(x,y,z)* do volume em coordenadas de visualização *(i,j,k)*. O mapeamento é realizado através da definição da matriz de projeção ortográfica, sendo efetuado de maneira mais eficiente através de cálculos incrementais. Assim, o passo incremental pode ser definido por:

$$\begin{pmatrix} \Delta i \\ \Delta j \\ \Delta k \end{pmatrix} = \begin{pmatrix} \Delta i/\Delta x & \Delta i/\Delta y & \Delta i/\Delta z \\ \Delta j/\Delta x & \Delta j/\Delta y & \Delta j/\Delta z \\ \Delta k/\Delta x & \Delta k/\Delta y & \Delta k/\Delta z \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix},$$

ou:

$$P_{i+\Delta i} = P_i + \Delta T \times \Delta i$$

O algoritmo pode então ler diretamente da matriz de transformação o tamanho dos incrementos de cada coordenada *(i,j,k)* para os incrementos de coordenadas *(x,y,z)*. Este procedimento é o inverso do realizado no lançamento incremental dos raios no algoritmo de *ray casting*. Os incrementos são constantes ao longo do volume, pois consideramos o *grid* uniforme com projeção ortográfica. Através da consulta ao sinal e à magnitude da mudança do valor de *k*, o algoritmo determina a ordem em que o volume será percorrido, de modo a garantir a ordem de composição desejada. O

eixo mais rápido é o que possui menor mudança em valores absolutos de *k* para cada passo ao longo de sua direção, e o sinal deste incremento define a extremidade inicial da composição. De maneira semelhante é definido o segundo eixo mais rápido e o terceiro. Os dois primeiros formam um plano que será projetado e terá suas contribuições acumuladas em um *buffer* denominado *sheet*. A ordem de composição (de frente para trás ou de trás para frente) pode ser definida pelo usuário, bastando para isso utilizar o processo de composição adequado.

Uma vez conhecidos esses valores e definida a ordem de caminhamento, o algoritmo transforma o ponto inicial com a utilização da matriz de projeção. Em seguida os demais pontos são transformados pelo esquema incremental. Os pontos transformados são então enviados para o processo de classificação/iluminação. Este processo determina a cor e a opacidade de acordo com o valor do *voxel* e aplica o modelo local de iluminação gerando a cor e a opacidade final do *voxel*.

O passo seguinte, central ao algoritmo, é a reconstrução. Nele o algoritmo se propõe a reconstruir um sinal contínuo, a partir de um volume discreto, de modo a reamostrar o sinal na resolução desejada para gerar a imagem.

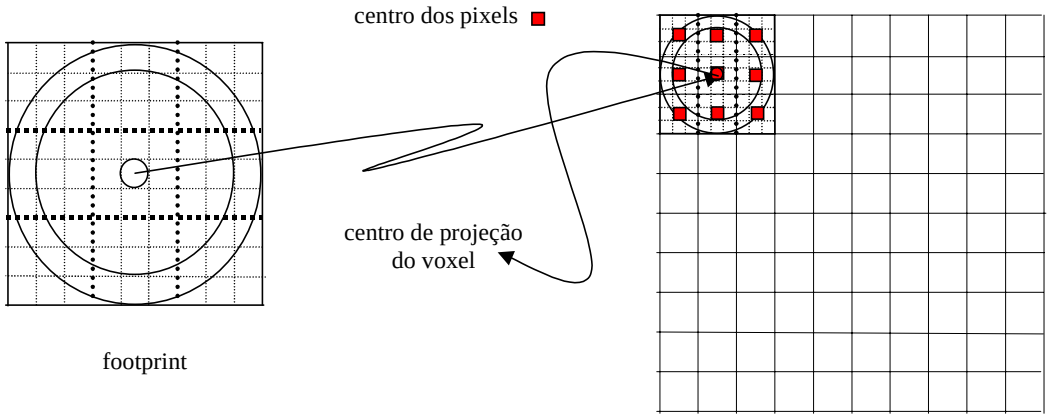


Figura 6.10 Utilização da tabela de f*ootprint*

Para isto é criada uma tabela representando a função de *footprint*, a qual é centralizada na posição de projeção de cada *voxel*. Em seguida determinam-se os *pixels* cujos centros estão contidos sob a projeção da tabela (os pontos ■ na Figura 6.10). Nessas posições pega-se o valor da tabela e ponderam-se os valores do *voxel* somando-os ao valor corrente no *sheet buffer*.

A forma mais simples de construir a tabela que representa a função de *footprint* é determinar a extensão da projeção do núcleo do filtro de reconstrução (*kernel*) sobre o plano da imagem e selecionar uma resolução maior que a da imagem

para amostrar a função. Em seguida, o núcleo de reconstrução é integrado para cada um dos pontos gerados.

Uma forma diferente de realizar este processo é através da modelagem do resultado com uma função simples (e.g. gaussiana), que é avaliada para cada ponto da tabela.

Para construir a tabela, o algoritmo calcula a extensão da projeção do núcleo de reconstrução e o mapeamento entre ela e a tabela normalizada básica gerada a partir de uma projeção ortográfica, sem operações de escala, da esfera unitária que contém o núcleo.

Para uma projeção ortográfica e um *grid* igualmente espaçado nas três direções, temos que o núcleo de reconstrução é mapeado em uma esfera. O algoritmo precisa criar uma tabela transformada apenas para levar em consideração as escalas uniformes que a transformação de visualização impôs aos *voxels*. Assim, a extensão da projeção é dada nas duas direções por $ext = 2 * filterwidth * gridscale * viewscale$ e o mapeamento é dado pela razão entre os raios das duas circunferências:

$$\text{(I)} \qquad mapping = \frac{1}{gridscale * viewscale}$$

O processo de visibilidade recebe a cor e a opacidade do *voxel* e pondera esses valores para gerar a contribuição em todos os *pixels* que estão sob a extensão do *footprint*. Os valores ponderados são compostos no *buffer* de acumulação, utilizando o esquema de acumulação apropriado à ordem de caminhamento no volume.

Como a reconstrução é um processo aditivo e a visibilidade não, ao finalizar a projeção de uma fatia do volume sobre o *sheet buffer*, realiza-se a etapa de composição deste *buffer* com a imagem já existente no *accumulation buffer*.

Um algoritmo semelhante, denominado *v-buffer*, foi apresentado em (Upson,1988). Esse algoritmo apresenta como diferença o fato de se basear em células e não em *voxels*. O algoritmo *v-buffer* percorre o interior da célula, interpolando os valores dos vértices e projetando cada valor interpolado no plano de visualização. Tanto o método de *splatting* como o *v-buffer* são facilmente paralelizáveis.

6.3 Mapeamento de Textura 3D

Este método surgiu basicamente como uma tentativa de utilização do *hardware* disponível para a aceleração do processo de *rendering*, mais

especificamente da capacidade de mapeamento de textura 3D realizada por *hardware*. Akeley (Akeley,1993) mencionou a possibilidade de utilização do *hardware* de mapeamento de textura para realizar o *rendering* de volumes ao apresentar a arquitetura e o desenvolvimento da máquina Reality Engine da Silicon Graphics. O algoritmo foi desenvolvido de forma independente por Cullip e Neuman (Cullip,1993), Guan e Lipes (Guan,1994), Cabral *et al.* (Cabral,1994) e Wilson *et al*. (Wilson,1994). Nos trabalhos de Cullip, Guan e Cabral, é necessário muito esforço de programação para calcular transformações, *clipping* e outros detalhes. Por isso, esta seção será baseada em (Wilson,1994), que deixa boa parte dessas tarefas para serem realizadas pela biblioteca gráfica, que efetivamente calcula as transformações e o *clipping* da textura.

Inicialmente, o algoritmo converte o volume de dados para um mapa 3D de texturas, usando um método de classificação baseado em tabelas que representam a função de transferência de cor e opacidade dos *voxels*. Os valores obtidos na classificação são modificados para levar em consideração o efeito da integração ao longo da profundidade (integral de *rendering*) e serem armazenados em um mapa de textura 3D. A modificação dos valores iniciais é baseada na integração dos valores dos *voxels* para a fatia do volume que será representada em uma fatia do mapa de textura. Este é aplicado aos planos que representam cada uma das fatias do volume, os quais são enviados para o *rendering*.

A conversão dos dados em mapas de textura é realizada apenas uma vez, sendo independente da posição do observador, precisando ser gerada novamente apenas quando acontecerem alterações nos dados ou nas funções de transferência.

O algoritmo pode ser definido com base em dois passos básicos: criação do mapa de textura e *rendering* das fatias.

O algoritmo interpreta cada plano a ser visualizado como localizado no centro de uma fatia do volume (espessura). Isto significa que cada plano de textura contribui com a cor e a opacidade correspondentes a uma fatia do volume de dados. Assim, sabendo a cor e a opacidade de um *voxel*, calculamos a cor e a opacidade que serão associadas àquele ponto no mapa de textura através de:

$$\alpha = \ln\left(\frac{1}{1-A_1}\right)$$

$$C = E\left(\frac{1-e^{-\alpha\Delta}}{\alpha\Delta}\right)$$

$$A = 1-e^{-\alpha\Delta},$$

onde *E* representa a cor do *voxel* (RGB*)* por unidade de comprimento, Δ é a espessura da fatia do volume, A_1 é a opacidade por unidade de comprimento e *C* e *A* são a cor e a opacidade associadas à textura. Uma dedução detalhada destas equações pode ser encontrada em (Whilhelms,1991). Estas equações representam o cálculo da integral de *rendering* ao longo da espessura da fatia do volume. Assim, a cada *texel* do volume de textura são associadas a cor *C* e a opacidade *A* apropriadas.

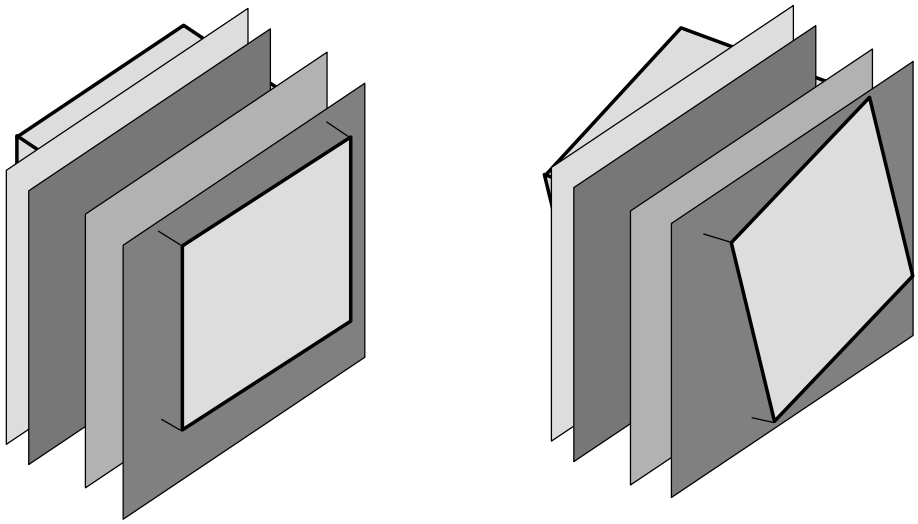


Figura 6.11 Planos para aplicação da textura.

Uma vez criada a textura 3D, é necessário aplicá-la a uma coleção de polígonos localizados em planos paralelos ao plano da imagem (Figura 6.11). Estes são clipados em relação ao volume de textura. O número de planos utilizados representa um conflito entre qualidade e velocidade. Quanto mais fatias são utilizadas, mais precisos os resultados e maior o tempo de resposta do algoritmo. O número de fatias exigidas para amostrar os dados adequadamente em uma dada direção é função da amostragem dos dados, conforme a freqüência de Nyquist.

Normalmente a memória de textura é menor que o volume de dados, sendo necessário dividir o volume em blocos pequenos, cada um definido como uma única

textura. Para preservar o ordenamento total das fatias, os blocos são ordenados e mapeados de acordo com o operador de composição apropriado.

6.4 Shear-Warp

O algoritmo *Shear-Warp* foi desenvolvido por Lacroute e Levoy (Lacroute,1995), utilizando o conceito de fatoração da matriz de visualização. Este é o mais popular algoritmo baseado em transformações afins do volume.

A base do algoritmo é a decomposição da transformação de projeção em duas etapas. Esta decomposição gera uma transformação de cisalhamento (*shear*) e uma de *warping*. O cisalhamento é usado para transformar o volume de modo que cada fatia fique paralela ao plano da imagem. Assim, os raios de visão percorrem de maneira trivial as fatias do volume, gerando uma imagem intermediária distorcida. A outra parcela resultante da decomposição é uma transformação de correção da deformação da imagem intermediária (*warp*), que gera a imagem final do objeto tridimensional. Estas duas fases podem ser facilmente visualizadas na Figura 6.12, que apresenta uma representação bidimensional deste processo.

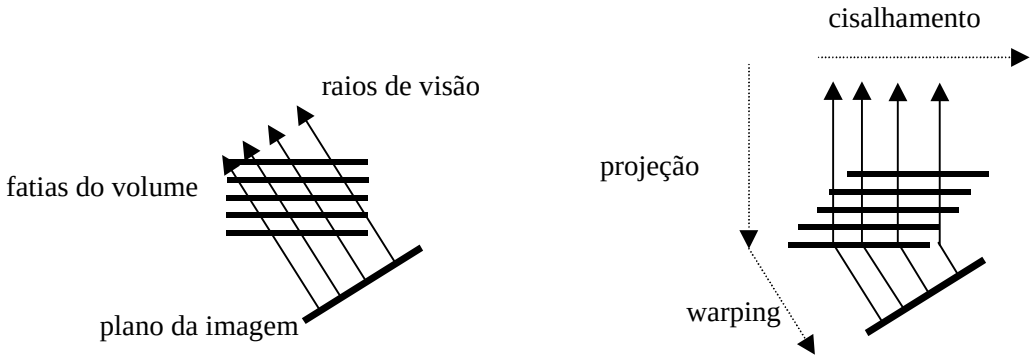


Figura 6.12 Esquema do algoritmo s*hear-warp*

O algoritmo pode ser decomposto em duas etapas básicas: construção da imagem intermediária e aplicação do *warping.*

Durante o processo de criação da imagem intermediária o volume é transformado para um sistema de coordenadas em que as fatias do volume são paralelas ao plano da imagem. A resolução da imagem intermediária depende das dimensões do volume, devendo ser tal que a projeção das fatias cisalhadas ocorra dentro desta imagem. Esta resolução pode ser calculada através da seguinte equação:

(II) $$ImgWidth = V_i + V_k * f_{shear}^{\ i}$$

(III) $$ImgHeight = V_j + V_k * f_{shear}^{\ j},$$

onde *ImgWidth* e *ImgHeight* representam a resolução da imagem intermediária e V_i , V_j e V_k representam as dimensões do volume. O fato da resolução da imagem intermediária estar relacionada com as dimensões do volume é importante para acelerar a etapa de composição dos valores dos *voxels*. Esta etapa é bastante sensível à introdução de *aliasing*.

```
TransformVolume (V)
ComputeImgSize (V)
para k de 1 a Vk
   para j de 1 a Vj
      para i de 1 a Vi
         se image[i,j] ≠opaco
            Composition (voxel[i,j,k], image[i, j])
         fim se
      fim para
   fim para
fim para
FinalImag = Warping (image)
```

Algoritmo 6.4 *Shear-Warping*

No Algoritmo 6.4 temos as etapas necessárias para a implementação deste algoritmo. A função *TransformVolume* aplica a transformação de cisalhamento de acordo com os parâmetros de câmera especificados pelo usuário. Em seguida é calculada a resolução da imagem intermediária e é iniciado o processo de acumulação das fatias do volume. Para cada *voxel* verificamos se a opacidade acumulada para o *pixel* correspondente é próxima de 1 (*pixel* opaco). Em caso negativo, realiza-se a composição do valor do *voxel* com o valor corrente do *pixel*. Ao final, após serem percorridos todos os *voxels*, aplicamos a transformação à imagem intermediária.

O algoritmo pode ser mais eficiente se utilizarmos uma estrutura do tipo RLE (*run length encoding*) para representar as *scanlines* de cada fatia do volume. Essa estrutura permite que os *voxels* transparentes não sejam processados. A representação RLE de uma linha de *voxels* é um conjunto de elementos formados por dois valores (*run* e valor associado). Na implementação proposta por (Lacroute,1995), essa estrutura é construída para representar os elementos não transparentes, permitindo que os conjuntos de *voxels* transparentes adjacentes em uma *scanline* da fatia do volume não sejam processados. Essa estrutura, que pode ser construída em um passo de pré-

processamento, utiliza seqüências de *voxels* não opacos seguidos de um deslocamento para a próxima posição que contém um *voxel* não opaco.

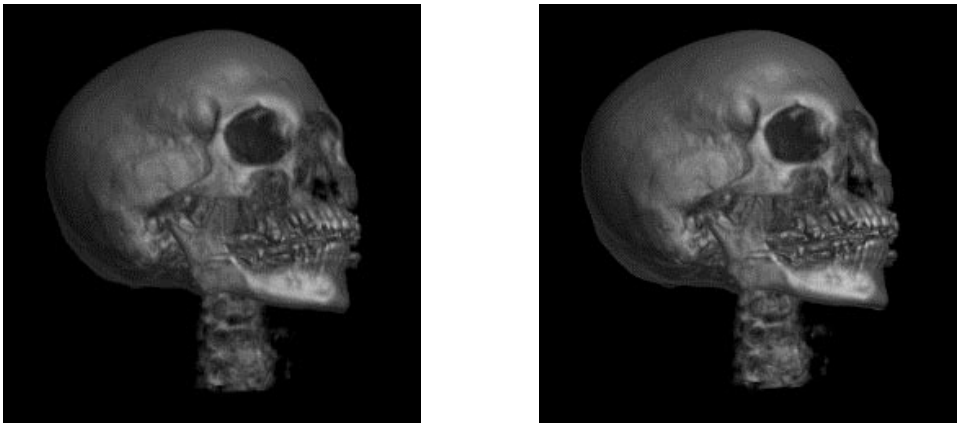
Se aplicarmos esta idéia de codificação em RLE para a imagem intermediária, podemos evitar o processamento de *voxels* associados a *pixels* já opacos na imagem. Assim, podemos aproveitar a coerência no espaço da imagem armazenando para cada *pixel* qual o próximo *pixel* que ainda não está opaco na mesma *scanline*. Isto é usado para evitar o processamento de *voxels* sobre *pixels* já opacos (seria o equivalente à terminação precoce do raio no algoritmo de *ray casting*). Esta estrutura é montada a medida que à imagem intermediária está sendo formada.

A combinação das duas estruturas RLE, juntamente com a propriedade de que as *scanlines* do volume estão alinhadas às da imagem intermediária, permite acelerar o procedimento de criação da imagem intermediária de duas formas:

- somente são processados *voxels* não-transparentes, ou seja, quando um *voxel* for transparente, passa-se para o cálculo da cor do *pixel* seguinte, pois o *voxel* atual nada tem a contribuir para a imagem;
- somente são processados *pixels* ainda não opacos, ou seja, quando um *pixel* acumular opacidade suficiente, tornando-se totalmente opaco, não há a necessidade de se continuar processando *voxels* para aquele *pixel*.

Assim, o método *Shear-Warp* só processa *voxels* não-transparentes enquanto o valor de opacidade acumulada não for considerado completamente opaco. Isto permite o aproveitar a coerência da imagem e do volume. Assim, esta implementação pode chegar a ser de 5 a 10 vezes mais rápida que o algoritmo de *ray casting* tradicional.

A Figura 6.13 mostra duas imagens, uma gerada com a utilização do método de *Shear-Warp* e outra através de *Ray Casting*. As duas imagens são de alta qualidade e satisfazem os requisitos de definição e confiabilidade necessários para a manipulação de exames médicos. A diferença é evidente quando observamos os tempos de processamento necessários para a geração de cada uma delas. Assim podemos concluir que o algoritmo *shear-warp* oferece maior interatividade que o *Ray Casting*.



Shear-warp (1.2s) Ray-casting (13.8s)

Figura 6.13 Comparação entre imagens geradas por *shear-warp* e *ray casting*

Após a composição completa do volume é necessário transformar a imagem intermediária para gerar a imagem final, o que pode ser realizado com um algoritmo bilinear de *warping*.

6.5 Métodos Baseados em Mudanças de Base

Nestes métodos, o volume é primeiramente transformado para outro domínio (Fourier, bases de cosseno, *wavelets*, etc.) e, a partir deste novo domínio, gera-se diretamente a visualização desejada. Os métodos baseados no domínio da freqüência utilizam o Teorema da Projeção de Fourier, o qual afirma que a projeção de um volume em uma dada direção pode ser obtida através da extração de uma fatia 2D perpendicular à direção de projeção e da aplicação da transformada inversa de Fourier a esta fatia. Esta abordagem obtém a projeção diretamente a partir do espectro 3D dos dados.

O objetivo inicial no desenvolvimento destes métodos foi uma tentativa de diminuir a complexidade do algoritmo de *rendering*. Posteriormente, foi descoberta a capacidade de utilização para grandes volumes de dados devido às características de compressão oferecidas pelas transformações utilizadas. Isto torna possível a visualização de dados volumétricos comprimidos sem exigir sua descompressão completa, o que reduz a necessidade de armazenamento, cálculo e sobrecargas de transmissão para aplicações em rede.

Nesta seção apresentamos uma visão geral dos algoritmos baseados na transformada de Fourier e de *wavelets*.

6.5.1 Domínio da Freqüência

A complexidade dos algoritmos de visualização de volumes é $O(n^3)$, pois todos os *voxels* precisam ser visitados para gerar a imagem. Embora existam alguns algoritmos eficientes para conjuntos de dados esparsos ((Levoy,1988), (Subramanian,1990), (Zuiderveld,1992)), essas otimizações são completamente dependentes dos dados, e a complexidade se mantém $O(n^3)$.

Como resultado de um esforço para reduzir drasticamente os custos de *rendering*, foram desenvolvidos algoritmos baseados no teorema de Projeção-Fatias de Fourier (*Fourier Projection-Slice*) ((Dunne,1990) e (Malzbender,1993)).

A idéia central desses métodos está associada à compreensão de que o problema de *rendering* de volumes pode ser visto como o problema inverso da reconstrução tomográfica. Nesta, o objetivo é calcular uma função de distribuição *f(x,y,z)* que gere um conjunto de projeções medidas. Por outro lado, em *rendering* de volumes, temos a função de distribuição e precisamos obter algumas de suas projeções.

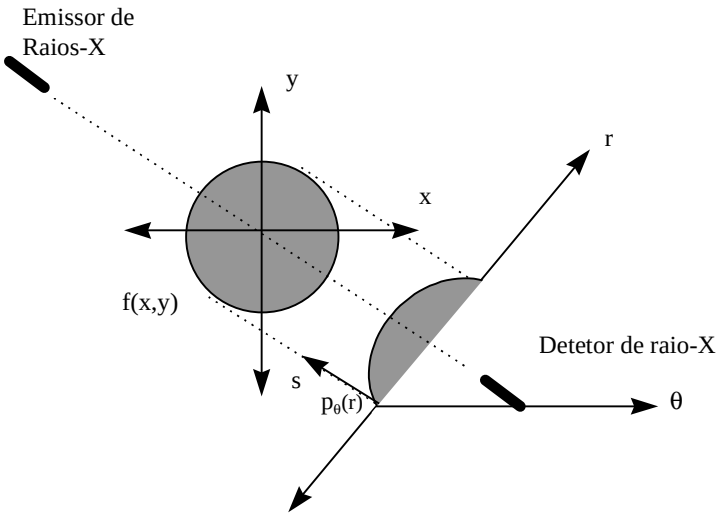


Figura 6.14 Geometria de aquisição de dados em tomografia computadorizada

No problema de reconstrução tomográfica, temos, inicialmente, um conjunto de projeções unidimensionais obtidas da distribuição de densidade (2D) que estamos tentando medir. Esses dados são adquiridos, por exemplo, através de um par emissor/detetor de raios X girando em torno da distribuição 2D que está sendo medida, conforme ilustra a Figura 6.14.

Isto é repetido para vários ângulos distintos e, em seguida, são utilizadas técnicas de reconstrução para calcular a função de distribuição de densidades da

amostra. Uma das formas de realizar esta tarefa é através do uso do Teorema Projeção-Fatia de Fourier ((Brigham,1974), (Cartwright,1990), (Dudgeon,1984)). Seja *F(u,v)* a transformada de Fourier da função *f(x,y)* e $P_{\Theta}(w)$ a transformada da projeção de *f(x,y)* denominada $p_{\Theta}(r)$, então o teorema afirma que:

$$P_{\theta}(w) = F(r\cos(\theta), rsen(\theta)).$$

Isto está ilustrado graficamente na Figura 6.15. Podemos enunciar o teorema da seguinte forma: "A transformada de Fourier 1D de uma projeção de um objeto 2D a um ângulo Θ é uma fatia da transformada de Fourier do objeto no mesmo angulo Θ".

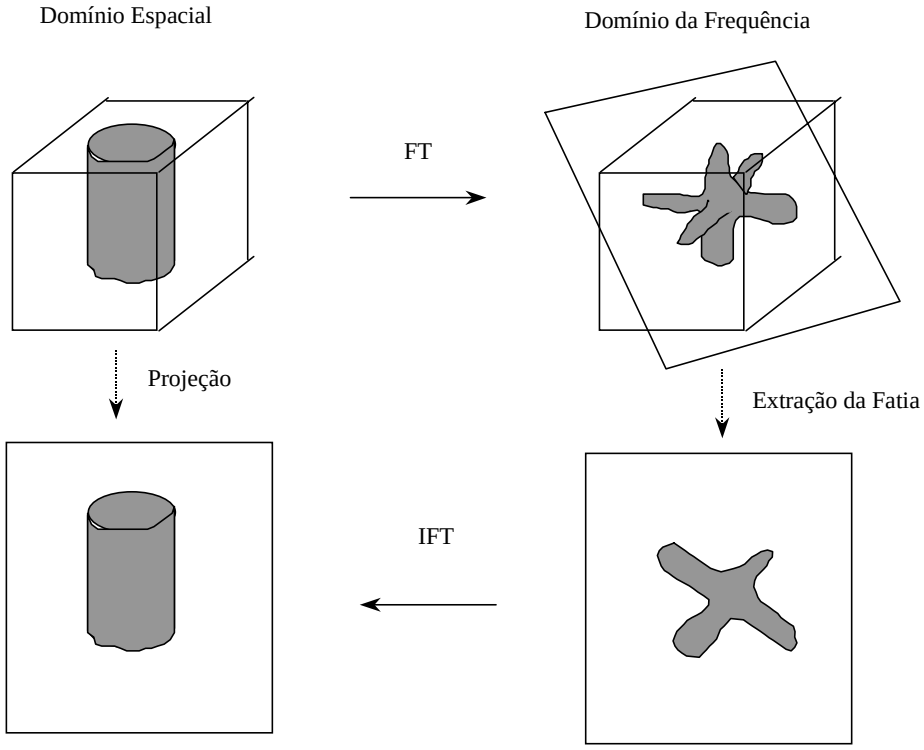


Figura 6.15 *Rendering* de volumes usando o teorema Projeção-Fatia de Fourier

Para um volume 3D, o teorema afirma que formam um par de transformadas de Fourier:

- a imagem 2D obtida através do cálculo de integrais de linha do volume ao longo de raios perpendiculares ao plano da imagem;
- o espectro 2D obtido pela extração de uma fatia da transformada de Fourier do volume, ao longo de um plano que inclui a origem e é paralelo ao plano da imagem.

Usando este teorema, uma vez gerada a transformada de Fourier do volume, uma imagem pode ser obtida, para qualquer direção de projeção ortográfica,através da extração de uma fatia 2D do espectro 3D, segundo uma orientação apropriada, e do cálculo de sua transformada inversa. O custo deste método é determinado pela Transformada Rápida Inversa de Fourier (TRIF), que tem complexidade $O(n^2 logn)$. Logo, verifica-se que este método possui uma vantagem teórica sobre os métodos que trabalham no domínio espacial, principalmente para volumes de grandes dimensões.

Embora possuam grande vantagem teórica, os métodos baseados na transformada de Fourier sofrem de vários problemas bem conhecidos:

- Alto Custo de Interpolação: dado que os pontos amostrados do espectro 3D e os da fatia 2D não coincidem, exceto na origem, é necessário interpolar e reamostrar o espectro 3D para poder extrair a fatia 2D. Como esta interpolação é imperfeita, temos que réplicas dos dados do volume não são completamente suprimidas, gerando uma imagem com a existência de fantasmas. Embora a interpolação seja $O(n^2)$, verifica-se que a constante associada à rotina de interpolação é alta, fazendo com que as implementações existentes percam grande parte do tempo de processamento realizando a interpolação, e tornando o algoritmo pouco atrativo para volumes de tamanhos comuns na prática (128^3 ou 256^3).
- Gasto de Memória: em função da utilização da aritmética complexa, associada à transformada de Fourier, é necessário manter em memória um par de números de ponto flutuante para cada *voxel*.
- Perda da informação de profundidade: a projeção obtida através deste método é uma integral de linha normal à direção de visada. Os *voxels* em um raio de visualização contribuem igualmente para a imagem, não influindo na sua distância ao observador. As imagens geradas perdem a informação acerca dos *voxels* que ocultam os posteriores.

Os dois primeiros problemas são de natureza técnica, existindo algumas soluções promissoras, como as propostas em (Takashi,1996). Já o problema da perda de informação de profundidade representa uma importante desvantagem do método. Entretanto, a oclusão não é a única indicação utilizada pelo sistema visual humano para determinar a forma e as relações espaciais entre os objetos. Outras indicações utilizadas são perspectiva, sombras, textura, atenuação atmosférica, visão

estereoscópica, acomodação ocular, etc. Assim, podemos aplicar o modelo de iluminação no domínio espacial antes de computar a transformada de Fourier do volume.

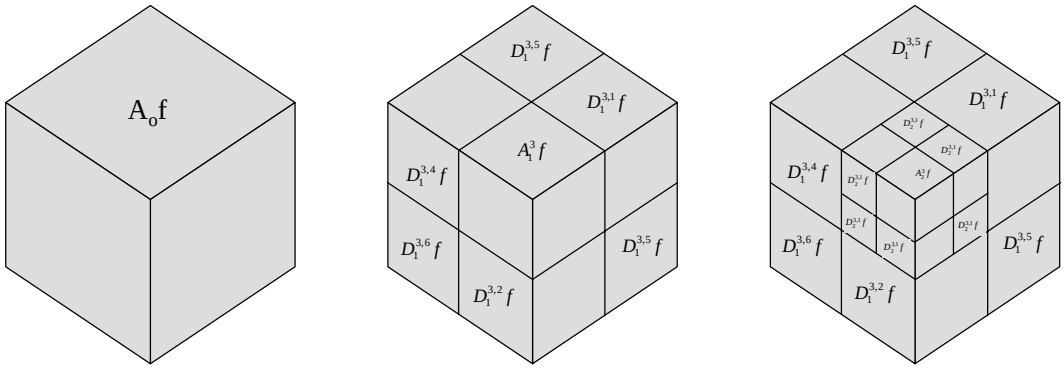
Para descrever o algoritmo, consideremos que *f(x)* é o volume e *F(u)* sua transformada de Fourier, com *x* e *s* sendo vetores 3D no domínio espacial e da freqüência. Dado *f(x)*, o algoritmo transforma-o para o domínio da freqüência para gerar *F(s)*. Isto é feito uma única vez. Para cada projeção do volume, interpolamos o espectro discreto ao longo do plano de extração (paralelo ao plano da imagem e passando pela origem) usando um filtro *H(s)*. O espectro interpolado é reamostrado para obter um espectro 2D, o qual é transformado inversamente para obter a projeção no domínio espacial.

Chiueh *et al.* (Chiueh,1994) dividem o volume em subcubos de tamanho *M x M x M*, gerando a transformada de Fourier de cada subcubo para comprimir os dados. A visualização é feita utilizando o teorema da Projeção-Fatia de Fourier em cada subcubo. Os efeitos de oclusão são obtidos através da utilização de um método de composição espacial das subimagens geradas, que apresenta complexidade $O(N^3 logM/M)$.

6.5.2 Domínio de Wavelets

A importância de bases de *wavelets* e análise em multiresolução advem da sua natureza hierárquica, que oferece uma abordagem matemática para a descrição dos diferentes níveis de resolução. Usando funções da base com boas propriedades de aproximação, podemos representar sinais somente com as caraterísticas importantes, desprezando as demais. Desta forma é possível obter representações comprimidas de funções, mantendo suas características gerais. Nesta seção apresentamos a visão geral de um algoritmo de *rendering* de volumes a partir dos coeficientes de *wavelet*. Detalhes acerca da teoria de *wavelets* podem ser encontrados em ((Daubechies,1992), (Chui,1992) e (Gomes,1998)).

A transformada de *wavelets* do volume gera uma representação hierárquica do volume (Figura 6.16). A cada passo da decomposição são geradas uma porção filtrada por um passa baixa (A_i f) e sete versões filtradas por um filtro passa banda (D_i f).



volume inicial primeira iteração segunda iteração

Figura 6.16 Volume transformado

O algoritmo de *rendering* baseado em *wavelets* deve resolver a integral de *rendering* usando uma representação em multiresolução da função escalar que representa o volume ((Gross,1995) e (Westermann,1994)). O algoritmo consiste dos seguintes passos:

- considere uma função escalar *f* representada por um volume de coeficientes de *wavelet* (Figura 6.16);
- use esses coeficientes para aproximar o valor de *f* em qualquer ponto *P(x,y,z)* do espaço (Figura 6.17);
- resolva a integral de *rendering* usando um algoritmo de *Ray Casting* padrão.

O primeiro passo está associado à avaliação da função *f* em um ponto no espaço utilzando a decomposição na base de *wavelets*. Começando no nível de resolução mais refinado, calcula-se a contribuição de todos os coeficientes de *wavelets* diferentes de zero e somam-se os valores obtidos à contribuição correspondente à função de escala. Neste *loop* verificamos qual o nível mais refinado no qual nenhum coeficiente de *wavelet* contribui para o valor da função. Se é encontrado um valor *j* diferente da resolução básica do volume, incrementamos o valor do passo de integração por um fator de 2^{j-1}, adaptando assim o passo de integração às informações contidas nos coeficientes. Esta etapa de caminhamento na estrutura de multiresolução é a etapa que mais consome esforço computacional. No caso geral, em que todos os coeficientes são diferentes de zero, são realizadas cerca de 100 multiplicações para interpolar um ponto.

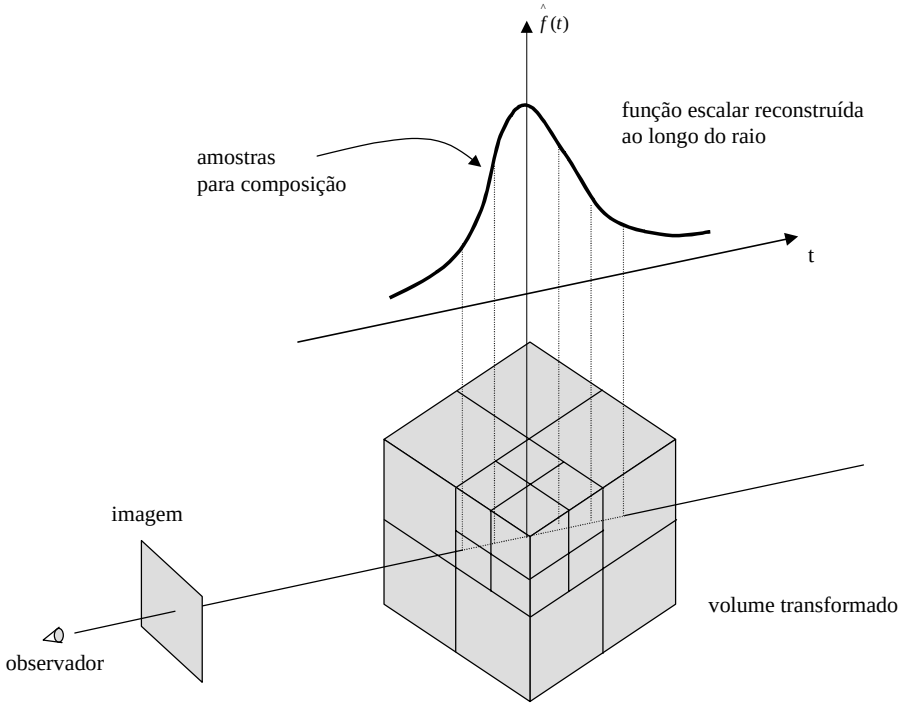


Figura 6.17 Lançamento do raio no volume de coeficientes

O último passo é a implementação do cálculo da integral de *rendering*, computando as contribuições advindas dos valores obtidos da estrutura de multiresolução. Várias técnicas de melhoria de desempenho do *Ray Casting* podem ser utilizadas, como a terminação precoce do raio e outras. O único aspecto novo neste algoritmo é a forma como o passo de integração é adaptado, através da utilização dos coeficientes de *wavelet*.

7. Referências

Artzy, E., Frieder,G., & Herman,G. "The Theory, Design, Implementation and Evaluation of a three Dimensional Surface Detection Algorithm", Computer Graphics and Image Processing, Vol. 15, No. 1, pp. 1-24, **1981**.

Aken, J. R. V., & Novak, M. "Curve Drawing Algorithms for Raster Display". ACM Transactions on Graphics, Vol. 4, No. 2, pp.147-169, April, **1985**.

Baker, H. H. "Building Surfaces of Evolution: The Weaving Wall", International Journal of Computer Vision, No. 3, pp. 55-71, **1989**.

Bentum, M. "Interactive Visualization of Volume Data". Ph.D. Thesis, University of

Twente, Enschede, The Netherlands, **1996**.

Blinn, J. "Models of Light Reflection for Computer Synthesized Pictures". Computer Graphics, Vol. 11, No**.** 2, pp. 192-198, **1977**.

Bloomenthal, J. "Polygonization of Implicit Surfaces", Computer Aided Geometrica Design, No. 5, pp. 341-355, **1988**.

Brigham, E. "The Fast Fourier Transform and Its Applications". Rev. 2nd ed., McGraw Hill, New York, pp. 385-410, **1986**.

Brooks, R. A. "Computational Principles of Transmission CT." In Medical Physics of Ct and Ultrasound: Tissue Imaging and Characterization, Edited by G. D. Fillerton and J. A. Zagzebski, American Institute of Physics, NewYork, **1980**.

Cabral, B., Cam. N., & Foran, J. "Acelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware". ?????????

Caldwell, J., Chowdhury, A., van Bemmel, P., Engelmark, F., Sonneland, L., Neidell, N. "Exploring for Stratigraphic Traps", *Oilfield Review*, Vol. 9, No. 4, pp. 48-61, **1997**.

Câmara, G., Battaiola, A., Oliveira, J.R., & Erthal, G. "Introdução à Visualização Científica", *CNMAC'91*, Minicurso do XIV CNMAC, **1991**.

Carr, J. "Surface Reconstruction in 3D Medical Imaging", Ph.D. Thesis, University of Canterbury, NZ, **1996**.

Cartwright, M. "Fourier Methods for Mathematicians Scietists and Engineers" Ellis Horwood, New York, **1990**.

Chen, L., *et al* "Surface Shading in a Cuberille Environment". IEEE Computer Graphics and Applications, Vol. 5, No.12, pp. 33-43, **1985**.

Chen, L. S. & Sontag, M. R. "Representation, Display, and Manipulation of 3D Digital Scenes and their Medical Applications". Computer Vision, Graphics and Image Processing, Vol. 48, No. 2, pp. 183-193, **1989**.

Chiueh, T.-C., He,T., Kaufman, A., & Pfister, H. "Compression Domain Volume Rendering". Technical Report TR.94.01.04, State University of New York at Stony Brook, **1994**.

Chui, C. K. "An Introduction to Wavelets". Academic Press, San Diego, 1992.

Cline, H., Lorensen, W., Ludke, S. Crawford, C., & Teeter, B. "Two Algorithms for Three-Dimensional Reconstruction of Tomograms", *Medical Physics*, Volume 15, Number 3, pp. 320-327, **1988**.

Coatrieux, J. & Barillot, C. "A Survey of 3D Display Techniques to Render Medical

Data". In: 3D Imaging in Medicine, Algorithms, Systems, Applications. Eds.: H. H. Hohne, H. Fuchs & S. M. Pizer, Springer Verlag, Berlin, **1990**.

Cohen, D. & Shaked, A. "Photo Realistic Imaging of Digital Terrain". Computer Graphics Forum, Vol. 12, No. 3, pp. 262-374, September, **1993**.

Cohen. D. "Voxel Traversal Along a 3D Line". GRAPHICS GEMS IV, Academic Press, pp. 366-369, **1994**.

Cullip, T. J., & Neuman, U. "Acelerating Volume Reconstruction with 3D Texture Hardware", Technical Report TR93-027, University of North Carolina, Chapel Hill, N. C., **1993**.

Daubechies, I. "Ten Lectures on Wavelets". SIAM Books, Philadelfia, PA., 1992.

Doi, A., & Koide, A. "An Efficiente Method of Triangulating Equivalued Surfaces by Using Tetrahedral Cells". IEICE Transactions on Comunications Eletronics Informations and Systems, Vol. E-74, pp. 214-224, **1991**.

Dorn, G.A., Cole, M.J. & Tubman, K. M. "Visualization in 3-D Seismic Interpretation", The Leading Edge, Vol. 14, No. 10, pp. 1045-1049, **1995**.

Doyle, M. "Quantization Effects in Digital Signals". 53rd Annual International Meeting, Society Exploration Geophysicists, Expanded Abstracts, pp. 510-513, **1983**.

Drebin, R.A., Carpenter, L., & Hanrahan, P. "Volume Rendering", *Computer Graphics*, Volume 22, Number 4, pp. 65-74, **1988**.

Dudgeon, D., & Mersereau, R. "Multidimensional Signal Processing". Prentice Hall, N. J., pp. 81, 82, 363-383, **1988**.

Duff, T. "Compositing 3-D Rendered Images ", Computer Graphics, Vol. 19, No. 3, pp. 41-44, July, 1985.

Dunne, S., Nappel, S. & Rutt, B. "Fast Reprojection of Volume Data". Proc. of the 1st Conference on Visualization in Biomedical Computing. Atlanta, GA, pp. 11-18, **1990**.

Dürst, M. J. "Additional Reference to Marching Cubes", Computer Graphics (ACM Siggraph Quarterly), Vol. 22, No. 2, **1988**.

Dyk, K. & Eisler, J.D. "A Study of the Influence of Background Noise on Reflection Picking", Geophysics, Vol. 16, No. 3, pp. 450-455, **1951**.

Ekoule, A. B., Peyrin, F. C., & Odet, C. L. "A Triangularizatin Algorithm from Arbitrary Shaped Muliple Planar Contours". ACM Transactions Graphics, Vol. 10, No. 2, pp. 182-199, April, **1991**.

El-Fallah, A.I., Ford, G.E., Algazi, V.R. & Estes Jr., R.R. “The Invariance of Edges and Corners Under Mean Curvature Diffusions of Images”, Proc. SPIE Image and Video Processing III, **1995**.

Elvins, T. T. “A Survey of Algorithms for Volume Visualization”. Computer Graphics, Vol. 26, No. 3, August, **1992**.

Foley, J.D., Van Dam, A. M., Feiner, S. K. & Hughes, J. F. “Computer Graphics: Principles and Practices’. Addison-Wesley, Reading, November, **1993**.

Fowler, J. E, & Yagel, R. “Lossless Compression of Volume Data”. ACM SIGGRAPH Symposium on Volume Visualization, pp. 43-50, **1994**.

Freitas, C.M.D.S, & Wagner, F.R. “A Methodology for Selecting Visual Representations in Scientific and Simulation Applications”, *SIBGRAPI’93*, Anais do VI SIBGRAPI, pp. 89-97, **1993**.

Frieder, G., Gordon, D., & Reynolds, R.A. “Back-to-Front Display of Voxel-Based Objects”, *IEEE Computer Graphics and Applications,* Vol. 5, No. 1, pp. 52-59, **1985**.

Fuchs, H., Kedem, Z.M., & Uselton, S.P. “Optimal Surface Reconstruction for Planar Contours”, Communications of the ACM, Vol. 20, No. 10, pp. 693-702, October, **1977**.

Gallagher, R., & Nagtegall, J. “An Efficient 3D Visualization Technique for Finite Element Models and Other Coarse Volumes”, Computer Graphics, Vol. 23, No. 3, pp. 185-194, **1989**.

Gattass, M., & ABEL, J.F. “Interactive-Adaptative, Large Displacement Analysis with Real-Time Computer Graphics”, Computer & Structures, Vol. 16, No. 14, pp. 141-152, **1983**.

Gelder, A.V., & Wilhelms J. “Topological Considerations in Isosurface Generation”, ACM Transactions on Graphics, Vol. 13, No.4, **1994**.

Glassner, A. S., editor “Graphics Gems V” , Normal Coding, pp. 257-264, Academic Press, New York, **1990**.

Gomes, J., & Velho, L. *Image Processing for Computer Graphics,* Springer-Verlag, New York, NY, **1997**.

Gomes, J., & Velho, L. *“From Fourier Analysis to Wavelets”*. *SIGGRPAH’98 Course Notes,* **1998**.

Gordon, D. & Reynolds, R. “Image Space Shading of 3-dimensional Objects”. Technical Report MIPG 85, Depto of Radiology, University of Pennsylvania, **1983**.

Gordon, D. & Reynolds, R. “Image Space Shading of 3-dimensional Objects”, Computer Vision, Graphics and Image Processing, Vol. 29, **1985**.

Gordon, D. & Udupa, J. K. “Fast Surface Tracking in Three-dimensional Binary Images”. Computer Vision, Graphics and Image Processing, Vol.45, No. 2, pp. 196-214, February, **1989**.

Goodsell, D. S., Mian, S. & Olson, A. J. “Rendering of Volumetric Data in Molecular Systems”. Journal of Molecular Graphics, Vol. 7, pp. 41-47, March, **1989**.

Gouraud, H. “Continuous Shading of Curved Surfaces”, IEEE Transactions of Computers, Vol. 20, No. 6, pp. 623-629, **1971**.

Gross, M. H., Koch, R., Lippert, L. & Dreger, A. “A New Method to Approximate the Volume Rendering Equation Using Wavelet Bases and Piecewise Polynomials”, Computer & Graphics, Vol. 19, No. 1, pp. 47-62, **1995**.

Guan, S., & Lipes, R. G. “Innovative Volume Rendering Using 3D Texture Mapping”. In SPIE Medical Imaging 1994: Images Captures, Foprmatting and Display. SPIE 2164, **1994**.

Hanrahan, P. “Three-pass Affine Transforms for Volume Rendering”, Computer Graphics, Vol. 24, No. 5, pp. 71-76, **1990**.

Heckbert, P. “Color Image Quantization for Frame Buffer Display”, Computer Graphics, Vol. 16, No. 3, pp. 297-304, **1982**.

Herman, G. T., & Udupa, J. K. “Display of 3-D Digital Images: Computational Foundations and Medical Applications”, IEEE Computer Graphics and Applications, Vol. 3, No. 5, pp. 39-46, **1983**.

Herman G.T., & Liu, H.K. “Three-Dimensional Display of Human Organs from Computer Tomograms”, Computer Graphics ans Image Processing, Vol. 9, No. 1, pp. 1-21, **1979**.

Herman, G.T. “Image reconstruction from Projections: The Fundamentals of Compputerized Tomography.”, Academic Press, New York, **1980**.

Hoehne, K.H., Bomans, M., Pommert, A., Riemer, M., Schiers, C., Tiede, U. & Wiebecke, G. “3D Visualization of Tomographic Volume Data Using the Generalized Voxel Model”, The Visual Computer, Vol. 6, No. 1, pp. 28-36, **1990**.

Jain, A. “Fundamentals of Digital Image Processing”, Prentice-Hall, New Jersey, **1989**.

Johnson, E. R., & Mosher, C. E. “Integration of Volume Rendering and Geometric Graphics”. Proceeding of Chapel Hill Workshop on Volume Visualization, Chapel

Hill, NC, pp. 1-8, May, **1989**.

Kaufman, A. "Memory and Processing Architecture for 3D Voxel-based Imagery", IEEE Computer Graphics and Applications, Vol. 8, No. 6, pp. 33-40, **1988**.

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

Kaufman, A., Yagel, R. & Cohen, D. "Intermixing Surface and Volume Rendering". In: 3D Imaging in Medicine, Algorithms, Systems, Applications. Eds.: H. H. Hohne, H. Fuchs and S. M. Pizer, Springer Verlag, Berlin, pp. 217-227, **1990**.

Kaufman, A. "Volume Visualization". Tutorial, IEEE Comp. Soc. Press, Los Alamitos, CA, **1991**.

Kalra, D. & Barr, A.H. "Guaranteed Ray Intersections with Implicit Surfaces", Computer Graphics (SIGGRAPH'89), Vol. 23, No. 3, pp. 297-306, **1989**.

Kalvin, A. D. "Segmentation and Surface Based Modeling of Objects in Three-Dimensional Biomedical Images", Ph.D. Thesis, New York University, **1991**.

Keppel, E. "Aproximating Complex Surfaces by Triangulation of Contour Lines". IBM Journal of Research and Development, Vol. 19, No. 1, pp. 45-75, January, **1975**.

Koide, A., Doi, A., & Kajioka, K. "Polyhedral Aproximation Approach to Molecular Orbit Graphics", Journal of Molecular Graphics, No.4, pp. 149-156, **1986**.

Krüger, W. "The Application of Transport Theory to Visualizationof 3D Scalar Data Fields". Computational Physics, Vol. 5, No. 4, pp. 397-406, **1991**.

Lacroute, P. & Levoy, M. "Fast Volume Rendering using a Shear-Warp Factorization of the Viewing Transformation", Computer Graphics, Vol. 28, No. 3, pp 451-458, **1995**.

Lacroute, P. "Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation", Ph.D. Thesis, Stanford University, **1995**.

Laur, D., & Hanrahan, P. "Hierarquical Splatting: A Progressive Refinement Algorithm for Volume Rendering", *Computer Graphics*, Vol. 25, No. 4, pp. 285-288, **1991**.

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

Levoy, M. "Efficient Ray-Tracing of Volume Visualization Algorithms", ACM Transactions on Graphics, Vol. 9, No. 3, pp. 245-261, **1990a**.

Levoy, M. "A Taxonomy of Volume Visualization Algorithms", *ACM*

SIGGRAPH'90, Course Notes, Course Number 11, pp. 6-11, **1990b**.

Levoy, M. "A Hybrid RayTracer for Rendering Polygon and Volume Data". IEEE Computer Graphics and Applications, Vol. 10, No. 3, pp. 33-40, March, **1990c**.

Levoy, M. "Viewing Algorithms." In: Kaufman, A., editor, *Volume Visualization*, IEEE Computer Society Press Tutorial, Los Alamitos, CA, pp. 89-92, **1991**.

Levoy, M. "Volume Rendering Using the Fourier-Projection Slice Theorem", Graphics Interface´92, pp. 61-69, **1992**.

Lorensen, W.E. & Cline, H.E. "Marching Cubes: A High Resolution 3D Surface Construction Algorithm", Computer Graphics (Proc. SIGGRAPH), Vol. 21, No. 4, pp 163-169, **1987**.

Luo, Y. & Schuster, G.T. "Wave Packet Transform and Data Compression", 62nd Annual International Meeting, Society of Exploration Geophysicists, Expanded Abstracts, pp. 1187-1190, **1992**.

Malzbender, T. & Kitson, F. "A Fourier Technique for Volume Rendering", *Focus on Scientific Visualization*, pp. 305-316, **1991**.

Malzbender, T. "Fourier Volume Rendering". *ACM Transactions on Graphics*, Vol. 12, No. 3, pp. 233-250, July, **1993**.

Max, N. "Optical Models for Direct Volume Rendering". IEEE Transactions on Visualization and Computer Graphics, Vol. 1, No. 2, pp. 99-108, July, **1993**.

Mccormick, B., Defanti, T. & Brown, M. "Visualization in Scientific Computing". Computer Graphics, Vol. 21, No. 6, November, **1987**.

McReynolds, T., Blythe, D., Fowle, C., Grantham, B., Hui, S. and Womack, P. "Programming with openGL: Advanced Rendering". SIGGRAPH ´97 Lecture Notes, Course 11, pp. 144-153, **1997**.

Meyers, D. Skinner, S. & Sloan, K. "Surfaces from contours ". ACM Transactions on Graphics, Vol. 11, No. 3, pp. 228-258, **1992**.

Muraki, S. "Volume Data and Wavelet Transform", IEEE Computer Graphics and Applications, Vol. 13, No. 4, pp. 50-56, July, **1993**.

Nielson, G. M. "Modeling and Visualizing Volumetric and Surface-on-Surface Data", *ACM SIGGRAPH'94*, Course Notes, Course Number 4, pp. 31-97, **1994**.

Ning, P. & Hesselink, L. "Vector Quantization for Volume Rendering". 1992 Workshop on Volume Visualization, pp. 69-74, October, **1992**.

Ning, P. & Hesselink, L. "Fast Volume Rendering of Compressed Data". Visualization 1993, pp. 11-18, **1993**.

Natarajan, B. K. “On Generating Topologically Correct Isosurface from Uniform Samples”, Tech. Report HPL-91-76, Software and Systems Lab., HP Company, 1991.

Neumann, U. “Volume Reconstruction And Parallel Rendering Algorithms: A Comparative Analysis”. Ph. D. Thesis, Computer Science Depto, University of North Carolina at Chapell Hill, 1993.

Nielson, G.M., & Hamann, B. “The Assymptotic Decider: Resolving the Ambiguity in Marching Cubes”, In Proceedings of Visualization’91, pp. 83-91, **1991**.

Nielson, G. M. “Scattered Data Modelling for Scientific Visualizations”. In: Course Notes Advanced Techniques for Scientific Visualization , SIGGRAPH’95, **1995**.

Oliveira Jr., P. P. M. & Gattass, M. “Detecção de Ativação em Ressonância Nuclear Magnética Funcional do Cérebro”, SIBGRAPI'97, **1997**.

Painter, S., Beresford, G. & Paterson, L. “On the Distribution of Seismic Reflection Coefficients and Seismic Amplitudes”, Geophysics, Vol. 60, No. 4, pp. 1187-1194, **1995**.

Phong, B. T. “Illumination for Computer Generated Pictures”, Communications of the ACM **,** Vol. 18, No. 6, pp. 311-317, **1975**.

Pommert, V., Tiede, U., Wiebecke, G., & Hoehne, K. H. “Surface Shading in Tomographic Volume Visualization: A Comparative Study”, Proceedings of the First Conference on Visualization in Biomedical Computing, Atlanta, GA, pp. 19-26, May, **1990**.

Porter, T. & Duff, T. “Compositing Digital Images”. Computer Graphics, Vo. 18, No. 3, July, **1984**

Reynolds, R. A. “Fast Methods for 3D Display of Medical Objects”. Ph. D. Dissertation, Depto. of Computer and Information Sciences, Univeristy of Pennsylvania, **1985**.

Rhodes, L. M. “Computer Graphics and Medicine: A Complex Partnership.” IEEE Computer Graphics and Applications, pp. 22-28, January, **1997**.

Robertson, P. K. “A Methodology for Choosing Data Representations”. IEEE Computer Graphics and Applications, Vol. 11, No. 3, pp. 56-67, May, **1991**.

Rosenblum, L.J. “Scientific Visualization at Research Laboratories”, Visualization in Scientific Computing, IEEE Computer Society Press, pp. 209-211, **1990**.

Sabella, P. “A Rendering Algorithm for Visualizing 3D Scalar Fields”, Computer Graphics, Vol. 22, No. 4, pp. 51-58, **1988**.

Sakas, G. & Walter, S. “Extracting Surfaces from Fuzzy 3D-Ultrasound Data.”, Computer Graphics (Proc. SIGGRAPH’95*),* pp*.* 465-474, **1995**.

Sakas, G., Schreyer, L. & Grimm, M. “Case Study: Pre-Processing, Segmenting and Volume Rendering 3D Ultrasonic Data.”, IEEE Computer Graphics and Applications, Vol. 15, No. 4, pp. 47-54, **1995**.

Sayood, K. “Introduction to Data Compression”, Morgan Kaufmann Publishers, San Francisco, CA, **1996**.

Seixas, R.B. “Visualização Volumétrica com Ray-Casting num Ambiente Distribuído”, Tese de Doutorado, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ, **1997**.

Seixas, R. B., Gattass, M., Figueiredo, L.H., & Martha, L.F. “Otimização do Algoritmo de Ray-Casting na Visualização de Tomografias”, *SIBGRAPI’94*, Comunicações do SIBGRAPI’94, **1994**.

Shirley, P. & Tuchman, A. “A Polygonal Approximation to Direct Scalar Volume Rendering”, Computer Graphics, Vol. 24, No. 5, pp. 63-70, **1990**.

Sousa, M.C. “Visualização Científica 3D da Simulação Numérica de Reservatórios de Petróleo”, Dissertação de Mestrado, Departamento de Informática, PUC-RJ, **1994**.

Speray, D. & Kennon, S. “Volumes Probes : Interactive Data Exploration on Arbitrary Grids”.Computer Graphics, Vol. 24, No. 5, pp. 1-12, **1990**.

Subramanian, K. R. & Fussel, D. “Aplying Space Subdivision Techniques to Volume Rendering”. Proceedings of the First IEEE Conference on Visualization(Visualization’90), IEEE Computer Society Press, pp. 150-159, **1990**.

Totsuka, T. & Levoy, M. “Frequency Domain Volume Rendering”. ACM SIGGRAPH’93, **1993**.

Tiede, U., Hoehne, K.H., Bomans, M., Pommert, A., Riemer, M. & Wiebcke, G. “Investigation of Medical 3D-Rendering Algorithms”, IEEE Computer Graphics and Applications, Vol. 10, No. 2, pp. 41-52, **1990**.

Tuy, H.K., & Tuy, L.T. “Direct 2D Display of 3D Objects”, IEEE Computer Graphics and Applications, Vol. 4, No. 10, pp. 29-33, **1984**.

Upson, C. & Keeler, M. “V-Buffer - Visible Volume Rendering”. Computer Graphics, Vol. 22, No. 4, August, **1988**.

Udupa,J.K., & Ajjanagade,V.G. “Boundary and Object Labelling in Three-Dimensional Images”, Computer Vision, Graphics and Image Processing, No. 51, pp. 355-369, **1990**.

Zuiderveld, K., Koning, A., and Viergever, M. “Acceleration of Ray-Casting Using 3D Distance Transforms”. Proceedings of the SPIE – Visualization in Biomedical Computing 1992, Vol. 1808, pp. 324-335, **1992**.

Yagel, R. “Volume Viewing: State of the Art Survey”. In Course Notes, SIGGRAPH’96.

Yeo, B. L. & Liu, B. “Volume Rendering of DCT-based Compressed 3D Scalar Data” IEEE Transactions on Visual Comput. Graph., Vol. 1, No. 1, pp. 29-43, March, **1995**.

Yagel, R., Cohen, D. & Kaufman, A. “Normal Estimation in 3D Discrete Space”, The Visual Computer, Vol. 8, No.. 5-6, pp. 278-291, **1992**.

Yeo, B. & Liu, B. “Volume Rendering of DCT-Based Compressed 3D Scalar Data, IEEE Transactions on Visualization and Computer Graphics, Vol. 1, No. 1, pp. 29-43, **1995**.

Watt, A., & Watt, M. “Advanced Animation and Rendering Techniques”, Addison-Wesley, **1993**.

Westerman, R. “A Multiresolution Framework for Volume Rendering”, 1994 Symposium on Volume Visualization, Washington, DC, pp. 51-58, October, **1994**.

Westermann, R. & Ertl, T. A. “Multiscale Approach to Integrated Volume Segmentation and Rendering”, Computer Graphics Forum (Proc. EUROGRAPHICS '97), Vol. 16, No. 3, pp. 117-129, **1997**.

Westermann, R. “A Multiresolution Framework for Volume Rendering”, Ph.D. Thesis, Universität Dortmund, Shaker Verlag Aachen, **1997**.

Westover, L. “Footprint Evaluation for Volume Rendering”, Computer Graphics, Vol. 24, No. 4, pp. 367-376, **1990**.

Westover, L. “Footprint Evaluation for Volume Rendering”, Computer Graphics (Proc. SIGGRAPH)**,** Vol 24, No. 4, pp. 367-376, **1990**.

Whilhelms, J. & Gelder, A. V. “A Coherent Projection Approach for Direct Volume Rendering”. Computer Graphics, Vol. 25, No. 4, July, **1991**.

Wilson, O., Gelder, A.V. & Wilhelms, J. “Direct Volume Rendering via 3D Textures”. Technical Report UCSC-CRL-94-19, Baskin Center for Computer Engineering and Information Sciences, University of California, Santa Cruz, **1994**.

Wolfe, R.H. & Liu, C.N. “Interactive Visualization of 3D Seismic Data: A Volumetric Method”, IEEE Computer Graphics and Applications, Vol. 8, No. 7, pp. 24-30, **1988**.

Wolberg, G. “Digital Image Warping”. IEEE Computer Society Press, Los Alamitos, CA, **1990**.

Wyvill, B., Mcpheeters, C. & Wyvill, G. “Data Structure for Soft Object”, The Visual Computer, Vol. 2, No. 4, pp. 227-234, **1986**.

Zucker, S.W. & Hummel, R.A. “A Three-Dimensional Edge Operator”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 3, No. 3, pp. 324-331, **1981**.