Visualização Tridimensional em Medicina usando Estereogramas Holográficos

Cesar Augusto de Carvalho Vannini

Dissertação de Mestrado

Instituto de Computação
Universidade Estadual de Campinas

Visualização Tridimensional em Medicina usando Estereogramas Holográficos

Cesar Augusto de Carvalho Vannini

9 de novembro de 2000

Banca Examinadora:

- Prof. Dr. Alexandre Xavier Falcão
  Universidade Estadual de Campinas
  Instituto de Computação. (Orientador)
- Prof. Dr. Roberto de A. Lotufo
  Universidade Estadual de Campinas
  Faculdade de Eng. Elétrica e Computação.
- Prof. Dr. Nelson Machado
  Universidade Estadual de Campinas
  Instituto de Computação.
- Prof. Dr. Paulo L. de Geus (Suplente)
  Universidade Estadual de Campinas
  Instituto de Computação.

Visualização Tridimensional em Medicina usando Estereogramas Holográficos

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Cesar Augusto de Carvalho Vannini e aprovada pela Banca Examinadora.

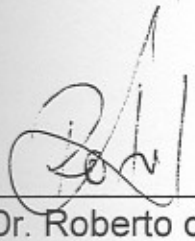
Campinas, 9 de novembro de 2000.

Prof. Dr. Alexandre Xavier Falcão
Universidade Estadual de Campinas
Instituto de Computação. (Orientador)

Prof. Dr. Cândido F.X. de Mendonça
Universidade Estadual de Maringá
Faculdade de Ciência da Computação.
(Co-orientador)

TERMO DE APROVAÇÃO

Tese defendida e aprovada em 22 de setembro de 2000, pela Banca Examinadora composta pelos Professores Doutores:



Prof. Dr. Roberto de Alencar Lotufo
FEEC – UNICAMP

Prof. Dr. Nelson Castro Machado
IC – UNICAMP

Prof. Dr. Alexandre Xavier Falcão
IC – UNICAMP

Agradecimentos

Agradeço em primeiro lugar a Deus, porque até aqui nos ajudou o Senhor (1 Sm 7:12b).
À minha esposa Vanice pelo apoio e paciência.
Às minhas filhas Isadora e Verônica pelo carinho e compreensão.
Ao meu orientador Alexandre, por ter acreditado em mim.
Ao meu co-orientador e irmão Xavier, por ter cumprido a ordenança do Senhor e ter colocado-me no caminho.
Ao Prof. Dr. José J. Lunazzi do Instituto de Física da Universidade Estadual de Campinas, por ter cedido a tela holográfica, sem a qual este trabalho não poderia ser realizado.
À CAPES pelo apoio financeiro.

Resumo

Apresentamos neste trabalho uma proposta que viabiliza a utilização de um equipamento, chamado holoprojetor 3.0, para visualização de imagens e animações tridimensionais em medicina. Esta proposta consiste da extensão da técnica shell rendering para codificação de estereogramas holográficos, os quais podem ser visualizados usando o holoprojetor 3.0 (ou através de óculos polarizados vermelho-azul).

Os estereogramas holográficos são formados pela codificação (rendering 2.5D) de três vistas de um objeto defasadas de um certo ângulo com paralaxe horizontal discreta. Cada uma das vistas está associada a um comprimento de onda, vermelho (R), verde (G) e azul (B), formando um quadro ou imagem 3D. Esta imagem 3D, ou uma seqüência de imagens 3D, é exibida em uma tela holográfica usando um projetor de vídeo RGB. O sistema completo, envolvendo desde a computação dos dados tomográficos até a visualização 3D na tela holográfica, foi denominado HoloView.

O sistema HoloView é a primeira ferramenta a proporcionar o uso interativo do holoprojetor 3.0. Experimentos apresentados neste trabalho mostram que o HoloView provê imagens de melhor qualidade do que os sistemas convencionais de visualização 2.5D de imagens médicas em um tempo de processamento razoável para computadores de baixo custo e poder computacional.

Abstract

In this work, we propose a solution that makes viable the use of an equipment, called holoprojector 3.0, to visualize 3D images and animations in medicine. Our purpose consists of extending the shell rendering technique for holographic stereograms codification, which can be visualized by the holoprojector 3.0 (or using red-blue polarized glasses).

Holographic stereograms are created by coding three views (2.5D renditions) of a same object with discrete horizontal parallax. Each view is assigned to one of the hues in the spectrum of colors, red (R), green (G), and blue (B), forming a 3D frame or image. An animation is created as a sequence of 3D frames like this. The holographic stereograms are exhibited on a holographic screen by using a RGB video projector. The whole system, involving the process from tomographic data computation to 3D visualization on the holographic screen, is denominated HoloView.

The HoloView is the first tool to allow interactive use of the holoprojector 3.0. Experiments, presented in this work, have shown that the HoloView provides images of better visual quality than those displayed by conventional 2.5D visualization systems. The computational time to generate such images is also found reasonable in low-cost low-powered computers.

Dedico este trabalho à minha esposa Vanice e às minhas filhas Isadora e Verônica por terem me suportado durante este tempo.

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Visualização tridimensional (3D) é uma área multidisciplinar que envolve física, engenharia e computação no estudo de metodologias para a exibição tridimensional de objetos de uma cena. Este processo requer o conhecimento do sistema visual humano e de seus indicadores de profundidade.

A visualização 3D encontra aplicações na engenharia, meteorologia, medicina, cristalografia de proteínas, modelagem molecular, biologia, etc. Neste trabalho, estamos interessados na visualização 3D em medicina.

1.1 Sistema Visual Humano

O sistema visual humano é o conjunto de órgãos do nosso corpo que nos faz enxergar. Constituído por um par de olhos e pelo cérebro, a função dos olhos é captar a cena com suas características, enquanto a função do cérebro é fazer o reconhecimento dos objetos contidos na cena. Uma parte muito importante da visão é a capacidade de reconhecimento da informação tridimensional através de indicadores de profundidade. A função de um sistema de visualização 3D é, portanto, reproduzir os indicadores de profundidade da melhor forma possível na imagem gerada. Estes indicadores são classificados em dois grupos.

Os indicadores do primeiro grupo são aqueles presentes em uma fotografia tradicional. Eles são observados em sistemas de visualização 2D, como monitores de computador e televisão, e podem ser captados pelo sistema visual humano ainda que este disponha apenas de um único olho. Considerando que apenas parte da visão 3D pode ser obtida com indicadores do primeiro grupo, dizemos que as imagens geradas nesses sistemas são 2.5D. São indicadores de profundidade do primeiro grupo:

Perspectiva linear: os objetos mais distantes na cena são vistos com dimensões

menores que os mais próximos.

Tonalização e sombreamento: a profundidade dos objetos na cena é captada pelo observador com base na posição relativa entre as fontes de luz e os objetos.

Perspectiva aérea: os objetos mais distantes são vistos com menos definição que os mais próximos.

Oclusão: os objetos mais proximos sobrepõem os objetos mais distantes que estão na mesma direção de visualização.

Gradiente de textura: a textura dos objetos mais distantes é menos definida que a dos objetos mais próximos.

Tamanho da imagem na retina: dois objetos de mesma dimensão conhecida pelo observador são vistos com tamanhos diferentes (o mais próximo maior que o mais distante) quando estão a distâncias diferentes.

Paralaxe temporal: vistas diferentes do objeto podem ser captadas através do movimento do observador ou do objeto.

Uma das motivações deste trabalho está no fato que os indicadores do primeiro grupo não são conclusivos para a percepção da informação tridimensional em uma cena. Pois, se nos basearmos em uma imagem contendo somente esses indicadores, podemos nos enganar quanto à observação dos objetos. Principalmente se os objetos forem desconhecidos ou não tiverem um referencial adequado. Neste contexto, estamos interessados em explorar também os indicadores do segundo grupo.

Os indicadores do segundo grupo são mais complexos, só podem ser reproduzidos em sistemas de visualização realmente 3D e, no caso do sistema visual humano, utilizando-se os dois olhos. Eles só podem ser representados no espaço 3D, onde as informações espaciais e temporais podem ser registradas de forma mais precisa, assim como encontramos na holografia tradicional. São indicadores do segundo grupo:

Acomodação: percepção do cérebro do observador para o ajuste focal dos olhos.

Convergência: rotação interna dos olhos focados no objeto à medida que o observador se aproxima ou se afasta do objeto.

Paralaxe binocular: cada olho do observador capta uma vista ligeiramente diferente da outra. Cerca de 5° de rotação uma da outra.

Paralaxe de movimento: a combinação de paralaxe temporal e paralaxe binocular.

1.2 Visualização 3D em Medicina

Em medicina, a visualização 3D tem como objetivo facilitar o entendimento de estruturas do corpo humano presentes em imagens tomográficas [11]. O diagnóstico por imagens, o planejamento de cirurgias [30] e radioterapias, a educação médica, o estudo de diversos fenômenos que alteram a anatomia e/ou função de órgãos do corpo humano, e a análise de movimento de articulações [29, 26] são exemplos de aplicações que requerem a visualização 3D como ferramenta central.

As imagens tomográficas são obtidas por equipamentos de raios-X (CT), de emissão de pósitrons (PET), de emissão de fótons (SPECT), ressonância magnética (MRI), ultrasom, e, no caso de estruturas biológicas, através de uma câmera confocal acoplada a um microscópio. Cada imagem tomográfica corresponde a uma imagem 2D de um corte transversal ao paciente. Uma seqüência de imagens tomográficas forma um volume de dados. Este volume de dados contêm objetos 3D (e.g. órgãos) que desejamos visualizar. Para tanto, técnicas de processamento de imagens são aplicadas para isolar estes objetos do resto dos dados. A simples projeção destes objetos 3D em um dispositivo 2D, tal como o monitor de um computador, possibilitaria apenas a visualização de imagens e animações 2D. Técnicas de computação gráfica (i.e. *rendering*) exploram indicadores do primeiro grupo, como tonalização, oclusão e paralaxe temporal, para transmitir uma impressão 3D dos objetos sobre o dispositivo 2D. Por esta razão, dizemos que a imagem gerada é 2.5D.

A maioria dos sistemas de visualização de imagens médicas exploram apenas os indicadores do primeiro grupo reproduzidos por técnicas de computação gráfica. Como exceção, existem os anaglifos, que exploram a paralaxe binocular, e os sistemas de realidade virtual, que exploram a paralaxe de movimento. Nos anaglifos, a percepção 3D é obtida com o uso de óculos com lentes vermelho-azul, para que cada olho veja apenas uma das duas vistas do objeto, ou olhar divergente, fixando os olhos em um ponto aquém ou além do plano do objeto. Nos sistemas de realidade virtual, o observador utiliza um óculos, ou capacete especial, e para cada movimento do observador, as imagens da cena apresentadas a cada olho devem ser atualizadas. Observe que nos sistemas baseados em técnicas de computação gráfica existe a desvantagem da imagem gerada ser 2.5D e nos outros dois existe desconforto para o observador que tem que usar algum tipo de dispositivo auxiliar ou truque de visão.

Portanto, visando propor um sistema de visualização realmente 3D em medicina, que não requer dispositivos auxiliares nem truques de visão, nós adotamos uma técnica de holografia baseada no conceito de `estereogramas holográficos`.

1.3 Estereogramas Holográficos

O conceito de estereogramas holográficos tem origem em duas técnicas de visualização 3D: estereografia e holografia, que são descritas em detalhes no Capítulo 3. Nesta seção, vamos apenas resumir o princípio de geração de um estereograma holográfico e seu processo de exibição.

Um estereograma holográfico é formado pela codificação (rendering 2.5D) de n vistas de um objeto defasadas de um ângulo θ (θ geralmente em torno de 3°) em paralaxe de movimento horizontal. Cada uma das vistas é associada a um comprimento de onda respeitando uma seqüência de valores crescentes (ou decrescentes) para formar um quadro ou imagem 3D. Esta imagem 3D é chamada estereograma holográfico. Sua exibição requer um mecanismo de projeção através de uma rede de difração, a qual separa e exibe as vistas no espaço 3D com paralaxe de movimento horizontal. Devido às limitações dos dispositivos disponíveis, o número de vistas utilizado é $n = 3$, cada uma correspondendo a uma cor, vermelho (R), verde (G) e azul (B). O mecanismo de projeção é um projetor de vídeo RGB e a rede de difração é chamada *tela holográfica* [23]. Este sistema de visualização 3D é descrito em detalhes no capítulo 3. A técnica utilizada é chamada `holoprojeção`.

1.4 Objetivo

A principal contribuição deste trabalho está em propor uma aplicação, no caso visualização 3D de imagens médicas, para o sistema de holoprojeção de estereogramas holográficos sugerido em [7]. Como resultado, o trabalho foi premiado com menção honrosa na conferência *SPIE on Medical Imaging, 1999* [9].

O objetivo deste trabalho é, portanto, mostrar a viabilidade de um sistema de holoprojeção de estereogramas holográficos para manipulação e visualização de imagens médicas. O sistema proposto é doravante chamado `HoloView`. Ele consiste de dois blocos de processamento. O primeiro é um sistema de visualização 2.5D baseado em técnicas de computação gráfica modificado para gerar estereogramas holográficos, e o segundo faz a exibição dos estereogramas por holoprojeção.

Considerando o fato que em aplicações médicas a interatividade é ponto crucial, a viabilidade do sistema holoview depende fundamentalmente de uma técnica rápida de rendering 2.5D. A técnica escolhida é denominada *shell rendering* [28]. Esta decisão se baseou no fato que recentemente a técnica foi considerada um dos métodos mais rápidos de rendering 2.5D, mesmo quando comparada a métodos que dependem de *hardwares* dedicados [19].

1.5 Organização do Texto

A apresentação deste trabalho inicia com um estudo sobre técnicas de processamento de imagens e computação gráfica utilizadas para visualização 2.5D em medicina (Capítulo 2). Como o conceito de estereogramas holográficos está associado à estereografia e à holografia, uma revisão sobre estas técnicas até o surgimento da holoprojeção é apresentada no Capítulo 3. O sistema HoloView é descrito em detalhes no Capítulo 4. Experimentos de manipulação e visualização com objetos de diferentes tamanhos foram realizados para verificar a interatividade do sistema e a qualidade das imagens geradas. Estes experimentos são descritos no Capítulo 5 e as conclusões e sugestões de continuidade deste trabalho são discutidas no Capítulo 6.

Capítulo 2

Visualização 2.5D em Medicina

2.1 Introdução

Definimos visualização 2.5D como o subcaso da visualização 3D formado pelas técnicas que exploram apenas os indicadores de profundidade do primeiro grupo. Entre estes, os indicadores mais explorados na área médica são perspectiva linear, tonalização, sombreamento, oclusão, e paralaxe temporal. Particularmente, estamos interessados na tonalização, oclusão e paralaxe temporal. Pois como veremos mais adiante, perspectiva linear e sombreamento não fazem sentido na visualização de estereogramas holográficos.

Em medicina, a classificação das técnicas de visualização 2.5D está associada ao processo de rendering da imagem. Existem várias possibilidades que serão comentadas na Seção 2.5. Inclusive técnicas que geram imagens 2D. Adotaremos, porém, o processo de geração de imagens 2.5D ilustrado na Figura 2.1. Este processo envolve quatro etapas: aquisição, interpolação, segmentação, e rendering 2.5D. A aquisição diz respeito ao processo de obtenção das imagens por tomografia. A interpolação e a segmentação requerem técnicas de processamento de imagens enquanto as técnicas de computação gráfica estão incluídas na etapa de rendering.

2.2 Aquisição

Imagens tomográficas são normalmente obtidas como cortes paralelos, aos eixos xy, xz e yz do paciente, e igualmente espaçados. A Figura 2.2a mostra imagens obtidas através de cortes feitos paralelos ao plano xy. Considerando as dimensões $p \times p$ dos pixels nessas imagens e o espaçamento d entre os cortes, cada pixel pode ser visto como um pequeno paralelepípedo de dimensões $p \times p \times d$ no espaço 3D (Figura 2.2b), dando origem ao conceito de `voxel` (i.e. *volume element*). O conjunto de todos os voxels definidos pela

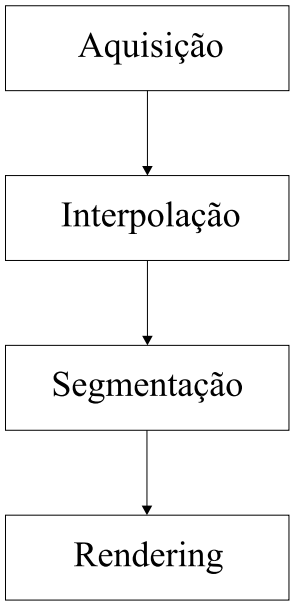


Figura 2.1: Etapas de processamento adotadas neste trabalho para a geração de uma imagem 2.5D.

seqüência de imagens tomográficas forma o que chamamos uma `cena 3D` (Figura 2.2c). Tipicamente, uma cena 3D contém cerca de 40 a 70 imagens tomográficas com dimensões 256×256 (ou 512×512) pixels cada.

Cada voxel em uma cena 3D tem associado um valor numérico correspondente a alguma propriedade física medida pelo dispositivo tomográfico. Por exemplo, na tomografia por Raios-X (CT), esta propriedade é a densidade dos tecidos. Sendo assim, quanto mais denso o tecido, mais claro ele aparece nas imagens. Por exemplo, o tecido ósseo aparece branco e tecidos moles aparecem escuros. Tipicamente, o valor numérico de um voxel varia no intervalo $[0, 255]$. Mas no caso de imagens de ressonância magnética (MRI), por exemplo, podem variar no intervalo $[0, 4096]$.

Um problema inicial, decorrente do processo de aquisição, está no fato que a dimensão d dos voxels é normalmente bem maior que a dimensão p (ver Figura 2.2b). Esta diferença causa distorções na imagem 2.5D gerada com o rendering (i.e. os objetos aparecem achatados). Para evitar essas distorções é crucial que os voxels tenham dimensões iguais em todas direções (i.e. $p = d$). A operação que transforma voxels não cúbicos em voxels cúbicos é chamada `interpolação`.

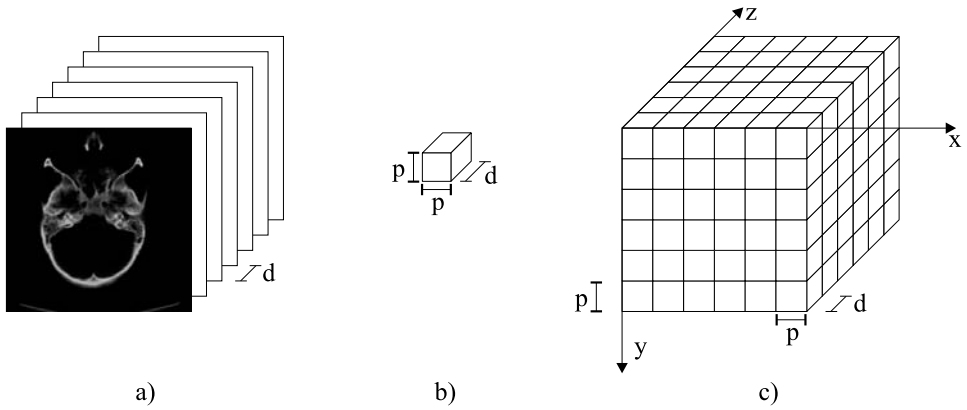


Figura 2.2: Formação da Cena 3D

2.3 Interpolação

A interpolação deve ser aplicada visando aumentar o número de imagens tomográficas da cena original [1]. Existem várias técnicas de interpolação [4], mas a mais usada é a `interpolação linear`.

Na interpolação linear, assume-se que os valores numéricos dos voxels variam linearmente ao longo de qualquer direção. No caso, estamos interessados na direção z (Figura 2.2). Portanto, considerando $v(x, y, z)$ e $v(x, y, z + d)$ dois voxels consecutivos ao longo do eixo z. A interpolação linear de um voxel $v(x, y, z + l)$, $l < d$, entre $v(x, y, z)$ e $v(x, y, z + d)$ é dada por:

$$I[v(x, y, z + l)] = \frac{(d - l) \times I[v(x, y, z)] + l \times I[v(x, y, z + d)]}{d}, \tag{2.1}$$

onde $I[v]$ é o valor numérico associado ao voxel v na cena 3D e $(x, y, z) \in R^3$.

Observe que podemos também aumentar a resolução da cena 3D nas três direções aplicando a interpolação linear ao longo de x, y, e z.

Um caso típico em medicina é uma cena 3D com $256 \times 256 \times 60$ voxels, onde $p = 0.60mm$ e $d = 1.0mm$. Neste caso temos $3.9M$ voxels. Se o valor numérico de cada voxel varia em $[0, 255]$, a cena 3D ocupa 3.9 Mbytes. Após interpolação, teremos uma cena 3D com $256 \times 256 \times 100$ voxels, onde $p = d = 0.60mm$, ocupando 6.5 Mbytes de memória. Mas no pior caso, uma cena 3D já interpolada pode chegar a cerca de 60 Mbytes de memória (ou melhor, $60M$ voxels). Isto reforça nossa motivação em encontrar um método rápido de rendering 2.5D.

Dada uma cena 3D, um problema alternativo, que pode ser tratado antes ou depois

[1] Poderíamos também reduzir o número de pixels de cada imagem tomográfica para obter $p' = d$, mas isto só pioraria a resolução e, consequentemente, a qualidade da imagem 2.5D desejada.

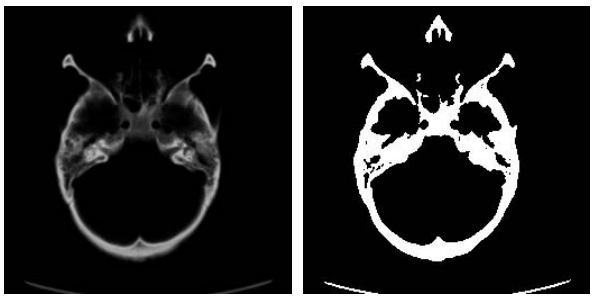


Figura 2.3: Segmentação por limiarização.

da interpolação, é `classificar` os voxels que pertencem a objetos de interesse. Esta operação é chamada `segmentação` de imagens.

2.4 Segmentação

Técnicas de segmentação de imagens podem ser classificadas em dois grupos: `automáticas` e `interativas`. As técnicas automáticas [5, 18] evitam a intervenção do usuário, mas seu sucesso nem sempre é garantido. Por isso, em muitas situações, técnicas interativas que procuram minimizar o esforço do usuário são adotadas [20, 13].

A segmentação de imagens é um problema complexo que não faz parte do objetivo principal deste trabalho. Assim, adotamos uma técnica simples de segmentação automática chamada `limiarização` (ou *thresholding*). A idéia da limiarização é prover um intervalo $[L, H]$ onde apenas voxels v com valor numérico $I[v] \in [L, H]$ são classificados como objeto de interesse. Na maioria dos casos de imagens de tomografia por Raios-X, por exemplo, onde os voxels têm associado um valor no intervalo $[0, 255]$, podemos classificar como tecido ósseo os voxels com valor $I[v] \in [175, 255]$. Mas é importante enfatizar que mesmo fixa a modalidade de aquisição, os valores de L e H podem variar para um mesmo tipo de tecido dependendo do protocolo de aquisição. Portanto, esses valores devem ser ajustados pelo usuário para cada cena 3D obtida. A Figura 2.3a, por exemplo, mostra uma imagem de tomografia por Raios-X de um crânio seco, ou seja, apenas a caixa óssea que encerra o cérebro utilizada para fins de estudo. Nesta imagem, o intervalo de limiarização $[63, 255]$ é escolhido para gerar a imagem binária do tecido ósseo apresentada na Figura 2.3b, onde os pixels brancos representam o objeto de interesse (osso) e os pixels pretos o fundo.

Para melhorar a qualidade da imagem sugere-se filtrar o resultado da segmentação, usando um filtro Gaussiano 2D [17] seguido de uma outra limiarização em $[110, 255]$. Isto elimina alguns ruídos nas bordas do objeto que resultam da primeira limiarização. Note que o procedimento é o mesmo aplicado para as outras imagens da cena 3D.

Em muitas situações é desejável aplicar a segmentação antes da interpolação. Exem-

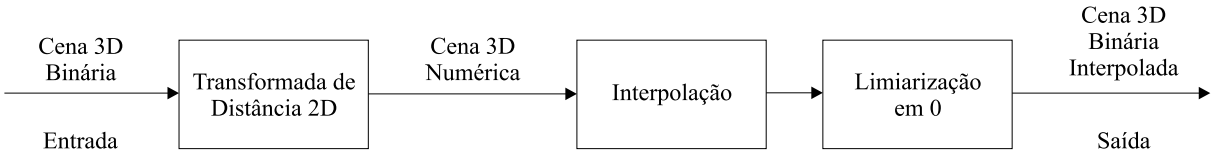


Figura 2.4: Fluxo de dados para a interpolação baseada na forma.

plos são os casos em que a segmentação interativa é adotada e casos em que os valores de limiarização já são conhecidos, portanto, a segmentação na cena 3D original implica em menos esforço computacional. Neste trabalho usamos as duas abordagens: segmentação seguida de interpolação e vice-versa.

A questão é como fazer a interpolação em uma cena 3D binária resultante da segmentação? A resposta a esta pergunta está na técnica de interpolação baseada na forma (*shape-based interpolation* [25]).

2.4.1 Interpolação Baseada na Forma

O processo de interpolação baseada na forma é ilustrado na Figura 2.4. A estratégia é converter inicialmente a cena 3D binária em uma cena 3D numérica. Esta conversão se baseia no cálculo da `transformada de distância 2D` que deve ser aplicada a cada imagem tomográfica da cena 3D binária. A transformada de distância associa a cada pixel de um objeto sua menor distância a um pixel da borda deste objeto [25]. Esta transformada também deve ser aplicada aos pixels do fundo. Neste caso, porém, a cada pixel do fundo deve ser associado o valor negativo da sua menor distância a um pixel de borda, levando em conta todas as bordas de objetos da imagem. Em seguida, a interpolação linear descrita na Seção 2.3, ou qualquer outra técnica similar, pode ser aplicada à cena 3D numérica. Uma cena 3D binária interpolada é finalmente gerada a partir da cena 3D numérica interpolada, por classificar como objeto os voxels com valor numérico não negativo e como fundo os demais.

Claramente, a transformada de distância está associada a uma métrica. Por uma questão de eficiência computacional, os algoritmos de interpolação baseada na forma usam normalmente a métrica de Chamfer [25] como aproximação da métrica Euclideana. A transformada de distância é também aplicada duas vezes, uma considerando os pixels de objeto e outra os pixels de fundo. Uma das contribuições neste trabalho é o uso do algoritmo baseado na transformada imagem-floresta (*image foresting transform* [12]) para calcular em tempo linear a transformada de distância Euclideana exata, considerando pixels de objeto e fundo simultaneamente. Maiores detalhes com relação ao método e ao algoritmo são descritos no Capítulo 4.

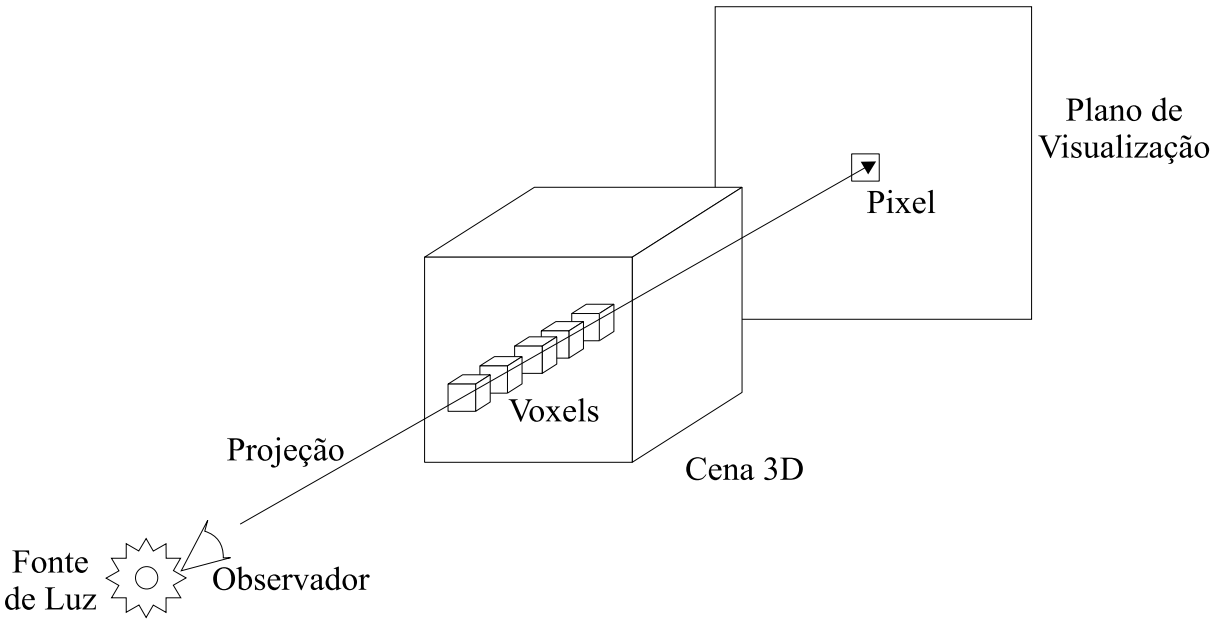


Figura 2.5: Projeção ortogonal dos voxels da cena 3D nos pixels do plano de visualização.

2.5 Rendering 2.5D

Entendemos por rendering uma transformação de uma cena 3D em uma imagem sobre um plano de visualização (Figura 2.5). Dizemos que o rendering é 2D, se o valor dos pixels da imagem gerada for igual ou uma combinação do valor de voxels da cena 3D. Dizemos que o rendering é 2.5D, se o valor projetado for calculado por uma função de tonalização que depende das coordenadas do voxel e do vetor normal à superfície de interesse que corta o voxel em questão.

Em medicina, existem várias técnicas de rendering 2D. Técnicas de `refatiamento`, por exemplo, se baseiam nas diferentes possibilidades de intersecção entre uma cena 3D (de preferência interpolada) com uma superfície, ou plano de corte, para gerar uma imagem 2D, ou uma nova seqüência de imagens tomográficas ao longo de um eixo arbitrário. Outro exemplo é a técnica de `projeção de intensidade máxima` (*maximum intensity projection*) que é bastante utilizada em cineangiografia. Nesta técnica, o valor projetado corresponde ao valor máximo entre os valores dos voxels que estão sobre uma mesma direção de visualização (ver Figura 2.5). Nesta seção, estamos interessados nas técnicas de rendering 2.5D e, em particular, nas técnicas de rendering 2.5D que requerem segmentação.

Técnicas de rendering 2.5D podem ser classificadas em dois grupos: `rendering de volumes` e `rendering de superfícies`. Em rendering de volumes [27, 22], a cena 3D interpolada é vista como um volume semi-transparente, no qual cada voxel tem um valor

de opacidade proporcional ao grau de interesse na visualização. Neste caso, não existe uma segmentação explícita para gerar uma cena 3D binária. Mas sim, um processo similar de classificação nebulosa (muitas vezes calculado durante a projeção) que atribui uma opacidade para cada voxel baseada no valor de cinza do voxel e nas características dos objetos com relação a este valor. Em rendering de superfícies [16, 21], a segmentação pode ser vista como uma classificação binária onde os voxels dos objetos são opacos e os do fundo são transparentes. Algumas destas técnicas, geram após a segmentação um modelo geométrico (normalmente baseado em triângulos) para a superfície dos objetos.

A abordagem utilizada neste trabalho pode ser vista como uma técnica mista, no sentido que requer segmentação como em rendering de superfícies mas armazena os voxels dos objetos sem criar nenhum modelo geométrico assim como em rendering de volumes. A técnica utilizada, chamada *shell rendering*, é descrita no Capítulo 4. Nesta seção, vamos apenas descrever os processos de projeção e tonalização de uma forma geral para cenas 3D binárias.

2.5.1 Projeção

Técnicas de projeção podem ser classificadas em dois grupos [11]: *voxels projection* e *ray casting*. Em *voxels projection*, os voxels visíveis pelo observador devem ser projetados da cena 3D para a imagem sobre o plano de visualização. Em *ray casting*, um raio de busca na direção de visualização parte de cada pixel da imagem para encontrar o voxel visível pelo observador na cena 3D. Neste trabalho, estamos interessados em *voxels projection*, porque é a técnica mais eficiente para *shell rendering*, como veremos no Capítulo 4.

A Figura 2.5 ilustra a projeção ortogonal (usando *voxels projection*) dos voxels de uma cena 3D no plano de visualização. Note que, a direção de visualização é ortogonal ao plano como se o observador estivesse a uma distância infinita da cena 3D. À medida que o observador se aproxima da cena, os raios de visualização deixam de ser paralelos e passam a ter um ponto de convergência (ou melhor, fulga) na posição do observador, ocorrendo então a projeção com perspectiva linear. Como veremos no Capítulo 3, a perspectiva linear é obtida pelo cérebro na holoprojeção. Portanto, os estereogramas holográficos devem ser gerados com projeção ortogonal.

Um aspecto importante na projeção é a técnica de remoção de voxels escondidos. Exemplos são *z-buffer*, *depth-sort*, *back-to-front* e *front-to-back* [4]. Neste trabalho, estamos interessados na técnica *front-to-back*, pois é a mais eficiente para o *shell rendering*, como veremos no Capítulo 4.

A idéia da projeção *front-to-back* [15] é simplesmente identificar em qual octante, dos oito possíveis octantes da cena 3D, incide a direção de visualização (ver Figura 2.6). Assumindo que a cena está entre o observador e o plano de visualização, como na Figura 2.5,

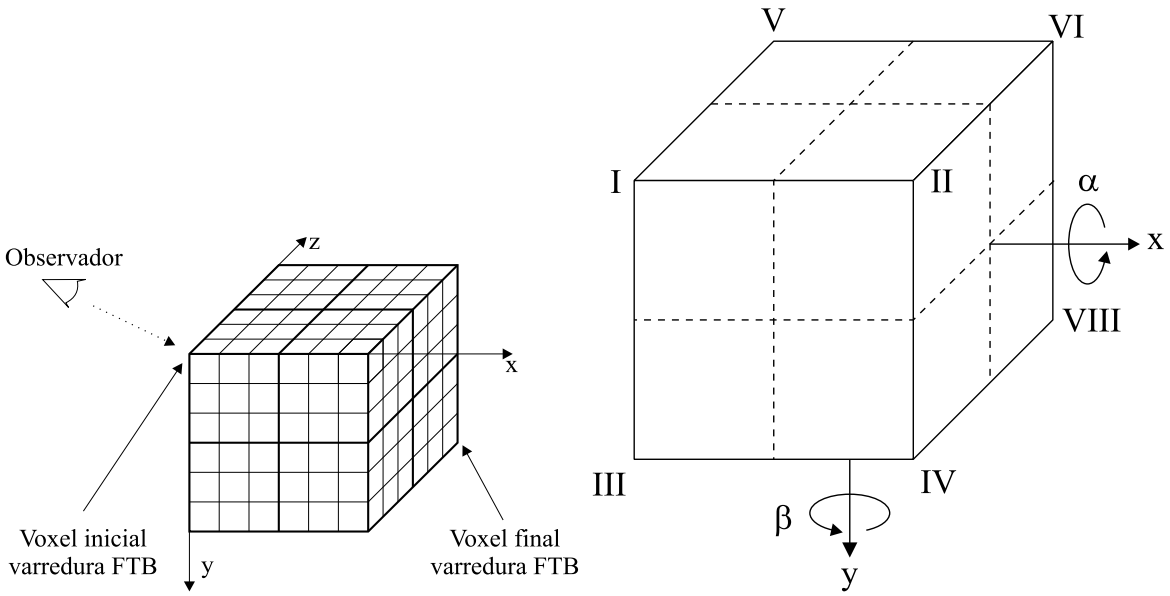


Figura 2.6: (a)Varredura FTB da cena 3D. (b) Octantes para acesso *front-to-back* aos voxels da cena 3D.

a Tabela 2.1 mostra a direção de varredura *front-to-back* dos voxels da cena 3D para cada octante. No capítulo 4, nós mostraremos como aplicar a regra desta tabela sobre a estrutura de dados usada no *shell rendering*.

2.5.2 Tonalização

A tonalização é a operação que associa a cada pixel $p(u, v)$ da imagem sobre o plano de visualização um valor de brilho. Este valor de brilho é resultado da simulação de como a luz refletida pelo ponto da superfície de um objeto, cujo voxel $v(x, y, z)$ correspondente está sendo projetado sobre o pixel $p(u, v)$, chega aos olhos do observador. Existem, portanto,

Caso de Acesso	Vértice Inicial	Vértice Final	Sentido de Indexação dos voxels
1	I	VIII	$x^- \to x^+, y^- \to y^+, z^- \to z^+$
2	II	VII	$x^- \to x^+, y^- \to y^+, z^+ \to z^-$
3	III	VI	$x^- \to x^+, y^+ \to y^-, z^- \to z^+$
4	IV	V	$x^+ \to x^-, y^- \to y^+, z^- \to z^+$
5	V	IV	$x^- \to x^+, y^+ \to y^-, z^+ \to z^-$
6	VI	III	$x^+ \to x^-, y^- \to y^+, z^+ \to z^-$
7	VII	II	$x^+ \to x^-, y^+ \to y^-, z^- \to z^+$
8	VIII	I	$x^+ \to x^-, y^+ \to y^-, z^+ \to z^-$

Tabela 2.1: Tabela de indexação dos voxels para acesso FTB

vários modelos de tonalização [14]. Neste trabalho adotamos o modelo de Phong.

$$T[p(u,v)] = T_{amb} + T_{dist}(u,v)(T_{dif}(u,v) + T_{esp}(u,v)), \tag{2.2}$$

onde:

- T_{amb} **é a tonalização ambiente:**

  Parcela de luz que ilumina o ambiente com tonalização constante.

  $$T_{amb} = T_{max} * K_a, \tag{2.3}$$

  onde:
  K_a é uma constante definida no intervalo $[0, 1]$ e T_{max} é o valor de brilho máximo desejado na imagem (e.g. 255).

- T_{dist} **é a tonalização por profundidade:**

  A intensidade de luz, refletida por cada voxel em direção ao observador, cai com o aumento da distância entre este voxel e o observador. Ou melhor, se considerarmos que a cena está entre o observador e o plano de visualização, podemos representar esta componente como uma função linear da distância entre o voxel projetado e o plano de visualização (Figura 2.5).

  $$T_{dist}(u,v) = \frac{T_{max} - T_{min}}{d_{min} - d_{max}} [d_{max} - d(u,v)] + T_{max}, \tag{2.4}$$

  onde:
  T_{max} é o valor de brilho máximo desejado na imagem (e.g. 255);
  T_{min} é o valor de brilho mínimo desejado na imagem (e.g. 0);
  d_{max} é a distância máxima entre um voxel e o plano de visualização;
  d_{min} é a distância mínima entre um voxel e o plano de visualização;
  d é a distância do voxel até o plano de visualização.

- T_{dif} **é a tonalização por reflexão difusa:**

  A reflexão difusa está associada à parcela de luz refletida na superfície de um objeto em todas as direções e que chega aos olhos do observador (Figura 2.7a). O valor de

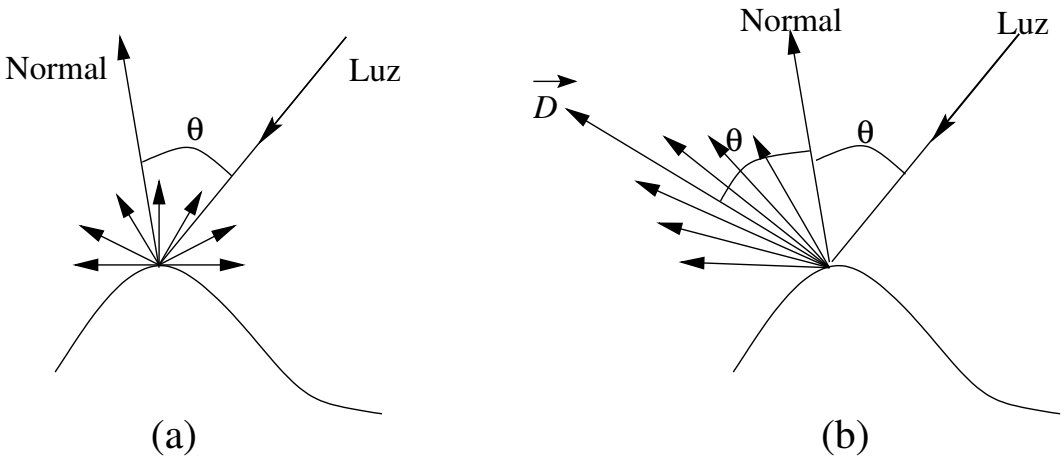


Figura 2.7: (a) Reflexão difusa. (b) Reflexão especular.

tonalização difusa varia com o cosseno do ângulo θ entre o vetor normal à superfície e a direção da luz incidente.

$$T_{dif}(u, v) = K_d \cos(\theta), \tag{2.5}$$

onde:
K_d é o coeficiente de reflexão difusa definido no intervalo $[0, 1]$.

- **T_{esp} é a tonalização por reflexão especular:**

  A reflexão especular está associada à parcela de luz refletida na superfície de um objeto na direção $\vec{D}$, que forma um ângulo $\alpha = 2\theta$ com a direção da luz incidente, e que chega aos olhos do observador (Figura 2.7b). O valor de tonalização especular varia com o cosseno do ângulo α.

$$T_{esp}(u, v) = K_e \cos^n(\alpha), \tag{2.6}$$

onde:
K_e é o coeficiente de reflexão especular definido em $[0, 1]$ e n um expoente cujo valor depende do material (e.g. $n = 2$).

Note que $K_a + K_d + K_e \leq 1$ é uma condição necessária se desejamos que a tonalização final $T[p(u, v)] \leq T_{max}$. Os ângulos θ e α requerem uma definição com relação à posição da fonte de luz, que no caso podemos assumir a mesma do observador (i.e. não ocorrem

sombras, assim como ilustrado na Figura 2.5), e o cálculo do vetor normal para cada ponto da superfície do objeto.

Existem várias formas de calcular o vetor normal em um voxel $v(x, y, z)$ de uma cena 3D. Uma alternativa é utilizar vizinhança-6 na cena 3D numérica resultante da interpolação linear da cena original.

$$N_x = \frac{I[v(x+1, y, z)] - I[v(x-1, y, z)]}{2}$$

$$N_y = \frac{I[v(x, y+1, z)] - I[v(x, y-1, z)]}{2}$$

$$N_z = \frac{I[v(x, y, z+1)] - I[v(x, y, z-1)]}{2}$$

$$N[v(x, y, z)] = \frac{(N_x, N_y, N_z)}{||(N_x, N_y, N_z)||} \tag{2.7}$$

onde:
N_x, N_y, N_z são as componentes do vetor normal, estimado no voxel $v(x, y, z)$, $(x, y, z) \in Z^3$;
$I[v(x, y, z)]$ é o brilho do voxel na cena original;
$N[v(x, y, z)]$ é o vetor gradiente, normalizado, no voxel.

O ângulo θ pode ser obtido através da componente N_z do vertor normal de cada voxel, conforme mostrado na Equação 2.8.

$$\theta = arccos(N_z) \tag{2.8}$$

2.5.3 Qualidade da Imagem Gerada

Para uma cena 3D típica $256 \times 256 \times 256$, sendo o valor dos voxels no intervalo $[0, 255]$, temos um conjunto de 16Mbytes ou $16M$ voxels. O tamanho de uma cena interpolada de alta qualidade pode chegar até muitas dezenas de Mbytes, o que constitui um problema em relação à velocidade do algoritmo de rendering. Para minorar este problema, podemos desconsiderar da cena 3D todos os voxels que não contribuem para a visualização, como por exemplo o fundo da cena e os voxels internos ao objeto de interesse. Esta é a motivação principal para termos adotado o método *shell rendering* na geração de estereogramas holográficos. A estrutura de dados *shell* armazena de forma eficiente apenas os voxels da casca que representa a superfície do objeto. Neste trabalho, estamos considerando cascas de espessura 1 voxel. Com isto, apenas 2% a 3% dos voxels da cena

são armazenados, representando uma economia considerável de tempo de processamento e espaço de memória.

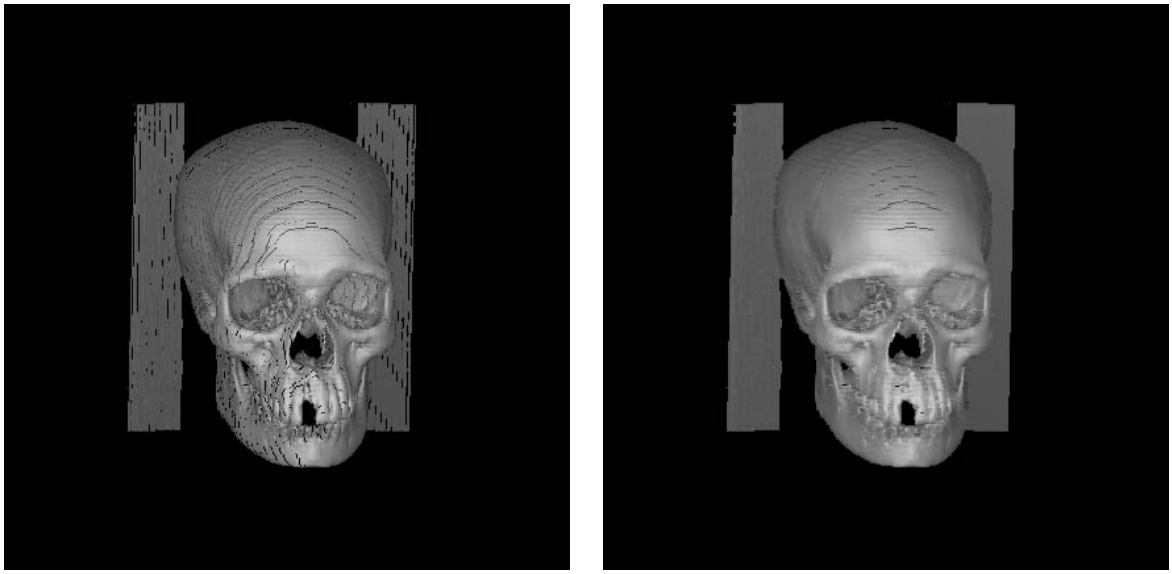
A classificação dos voxels para a criação do shell de espessura 1 será discutida com detalhes no Capítulo 4.

Devido ao fato do *shell* possuir apenas 1 voxel de espessura, alguns buracos são visíveis na imagem renderizada atrapalhando o estudo da imagem, principalmente no caso 3D. Este problema pode ser resolvido aplicando-se um filtro na imagem renderizada, como por exemplo fechamento morfológico seguido de reconstrução por erosão, entre outros. A Figura 2.8a mostra o rendering de superfícies aplicado sobre um shell com 1 voxel de espessura sem nenhum filtro de fechamento de buracos. A Figura 2.8b demonstra a ação de um filtro menos deformante, que na maioria dos casos resulta num fechamento quase total dos buracos, e na Figura 2.8c podemos ver a ação de um filtro potente que apesar de fechar todos os buracos provoca uma leve deformação na imagem (i.e. suavização dos detalhes finos).

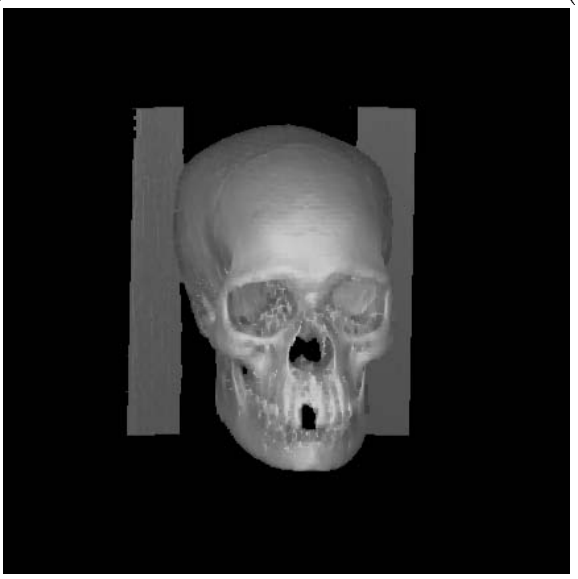
2.6 Conclusão

O rendering de imagens médicas exige um compromisso entre tempo de processamento e qualidade da imagem de saída. A técnica utilizada neste trabalho, denominada *shell rendering*, pode ser considerada como uma abordagem mista entre técnicas de rendering de volumes e de superfícies. O resultado obtido é tipicamente uma imagem 2.5D como pode ser visto na Figura 2.9.

As imagens geradas com as técnicas convencionais de rendering exploram os indicadores de profundidade apenas do primeiro grupo, gerando imagens consideradas 2.5D devido à impressão 3D conseguida. Uma maneira de melhorar a codificação da informação 3D na imagem gerada, seria modificar o algoritmo de rendering para gerar estereogramas holográficos digitais, explorando assim também alguns indicadores de profundidade do segundo grupo. Estes estereogramas podem ser vistos em 3D real com o auxílio de óculos estéreos (vermelho-azul) ou projetando-se o estereograma sobre uma tela holográfica. Denomina-se holoprojeção a técnica de projetar estereogramas hologáficos sobre uma tela holográfica, criando-se um sistema de visualização auto-estereoscópico. Os próximos capítulos abordam com detalhes a implementação do Holoprojetor e a geração dos estereogramas holográficos digitais.



(a) (b)



(c)

Figura 2.8: (a) Imagem original de rendering de superfície. (b) Resultado de filtro parcial para remoção de buracos. (c) Remoção total de buracos.

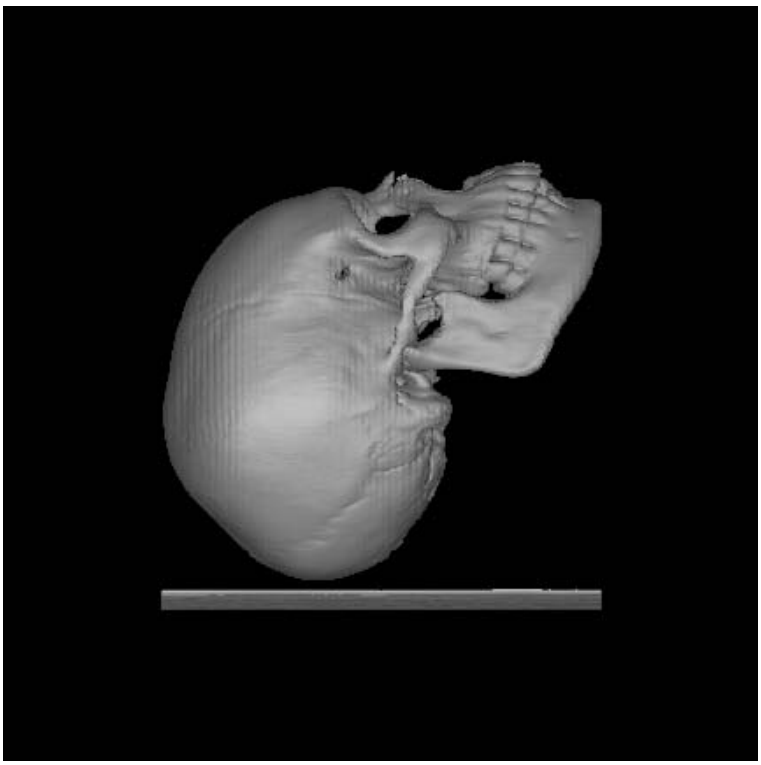


Figura 2.9: Rendering 2.5D de um crânio seco.

Capítulo 3

Estereogramas Holográficos

3.1 Introdução

Técnicas de visualização 3D que exploram os indicadores de profundidade do primeiro e segundo grupos podem ser classificadas em: `estereográficas` (ou `estereocópicas`) e `holográficas`. A estereografia é composta por técnicas que exploram a paralaxe binocular (i.e. técnicas que mostram a cada olho do observador uma imagem diferente). A holografia explora todos, ou quase todos, indicadores de profundidade e se divide em: `tradicional` (ou `estática`) e `dinâmica`. A diferença básica entre elas é que a holografia tradicional só reproduz imagens 3D estáticas, enquanto a holografia dinâmica consegue gerar animações 3D.

Os estereogramas holográficos resultam de uma combinação entre técnicas estereográficas e uma técnica de holografia dinâmica chamada holoprojeção. Este capítulo, portanto, descreve uma revisão dessas técnicas até estereogramas holográficos.

3.2 Estereografia

A estereografia ou estereoscopia consiste em apresentar aos olhos do observador vistas diferentes da mesma cena, requerendo que o observador utilize acessórios visuais ou truques de visão para enxergar a cena em 3D. Dentro de estereografia podemos citar como suas principais técnicas: `par estéreo`, `estereograma tradicional`, `anaglifos` e `realidade virtual`.

3.2.1 Par Estéreo

A técnica de par estéreo consiste em posicionar duas imagens lado a lado, onde cada uma representa uma vista diferente da mesma cena (cerca de cinco graus de diferença e

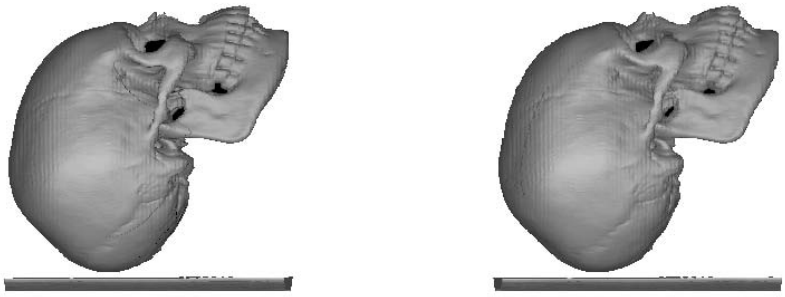


Figura 3.1: Par estéreo para visualização convergente.

oito centímetros de afastamento). A imagem em 3D pode ser vista utilizando-se técnicas de olhar convergente ou divergente. Algumas pessoas podem apresentar dificuldades ou incapacidade para observar as imagens em 3D devido à necessidade de um grande controle ocular para efetuar o olhar divergente ou convergente.

A Figura 3.1 mostra um exemplo de cena 3D em par estéreo. Para observar corretamente esta cena o observador deve utilizar a técnica de olhar convergente.

3.2.2 Estereograma Tradicional

Estereogramas tradicionais, também conhecidos por `estereogramas de imagem única` ou `auto-estereogramas` têm sua origem em estudos científicos sobre a visão. A técnica consiste em combinar duas imagens em apenas uma utilizando-se imagens de pontos aleatórios gerados por computador. O observador pode utilizar as técnicas de visão convergente ou divergente para observar a imagem 3D. Imagens deste tipo estão presentes em alguns livros de "mágica" (ilusões ópticas), por criarem ilusões tridimensionais à partir de uma figura de formas abstratas. Podemos citar como exemplos de técnicas de estereogramas: estereograma de imagens únicas de pontos aleatórios (SIRDS), estereograma mapeado em imagens, estereograma de texto normal de imagem única (SINTS), estereograma de texto aleatório de imagem única (SIRTS) e estereograma baseado em ícones.

3.2.3 Anaglifos

Um anaglifo consiste na combinação de duas vistas diferentes de uma cena na mesma imagem, utilizando-se cores distintas para cada uma das vistas como vermelho-azul ou vermelho-verde. Para a observação da cena 3D o observador é obrigado a utilizar óculos estéreos vermelho-azul para separar as diferentes vistas. O uso destes óculos pode criar um certo desconforto para o observador, sendo que as imagens são vistas em tons de roxo ou amarelo, dependendo da combinação de cores utilizada, ou a aparição de "imagens fantasmas" devido a uma filtragem não perfeita das vistas mediante a baixa qualidade dos filtros do óculos.

Existem algumas limitações nesta técnica como por exemplo: limites de fusão máximos entre 24.0 e 27.0 minutos de arco para a dispariadade binocular e limitações do número de vistas. Como conseqüência destas limitações, a posição dos objetos na cena é fixa (sem paralaxe) e a profundidade é limitada. Anaglifos coloridos podem ser criados, porém não podendo ser utilizados para todas as imagens devido ao fato de que em algumas combinações de cores o resultado desejado não pode ser produzido.

3.2.4 Realidade Virtual

É possível que alguém defenda a idéia de que realidade virtual não é somente estereografia, visto que esta utiliza uma combinação da paralaxe temporal com a paralaxe binocular, entretanto, tal sistema requer a detecção da posição do observador a todo instante para produzir as vistas baseadas nesta nova referência. Cada uma destas vistas é apresentada separadamente aos olhos do observador pelo uso de óculos de realidade virtual ou capacete.

3.3 Holografia

Holografia consiste em registrar todas ou quase todas as informações tridimensionais de uma cena (*holos* em grego significa todo). A `holografia tradicinal` utiliza-se de um filme especial para tirar uma "fotografia 3D" da cena, enquanto que a `holografia dinâmica` utiliza um computador para gerar partes do processo de registro e reconstrução da cena 3D.

3.3.1 Holografia Tradicional

Holografia tradicional consiste em registrar a informação tridimensional de um objeto sobre um filme de alta resolução chamado `holograma`. Esta é uma das técnicas mais antigas de visualização 3D e a que apresenta um dos melhores resultados finais, contendo todos os indicadores de profundidade do primeiro e segundo grupos.

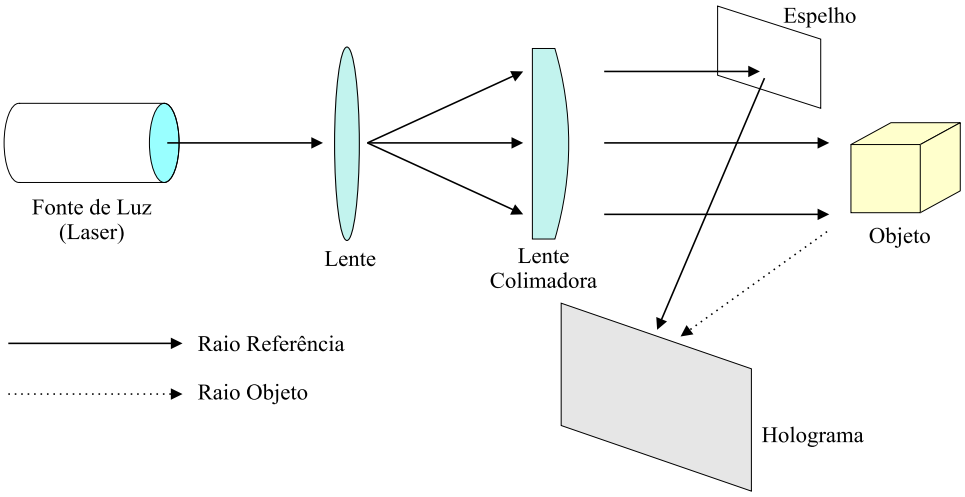


Figura 3.2: Processo de registro da holografia.

Um holograma é capaz de armazenar as franjas de interferência da luz refletida por um objeto tomando como base uma luz de referência. Na realidade um holograma não contém uma imagem, mas sim um conjunto grande de micro espelhos (chamados franjas de interferência) que refletem luz em um dado intervalo espectral. Dizemos que o holograma pode ser reconstruído quando devidamente iluminado, pois podemos observar a luz refletida pelo conjunto de micro espelhos. Este processo, portanto, se divide em duas etapas: `registro` e `reconstrução`.

O registro utiliza uma fonte de luz, que por meio de um arranjo óptico de lentes, é dividida em dois raios de luz. Um dos raios, chamado `raio de referência`, é conduzido na sua totalidade para o holograma, de maneira a registrar a referência luminosa a que o objeto está sendo exposto (Figura 3.2). O segundo raio é direcionado para o objeto em questão. A luz refletida pelo objeto é então registrada junto à informação de referência no holograma. Esta informação é registrada como franjas de interferência luminosa, onde cada espectro de luz representa uma vista do objeto.

A reconstrução da imagem é feita pela iluminação do holograma com uma luz similar à luz utilizada no registro. Desta maneira podemos visualizar a informação 3D registrada e ao movermos o holograma veremos diferentes vistas do mesmo objeto. Maiores detalhes quanto aos processos de registro e reconstrução de hologramas podem ser vistos em [2].

Uma desvantagem da holografia tradicional é que um holograma mostra uma imagem 3D estática do objeto, denominada *hardcopy*.

3.3.2 Holografia Dinâmica

A idéia básica na holografia dinâmica é gerar animações 3D com todos, ou quase todos, os indicadores de profundidade, sem a utilização de hologramas. Para tanto, o computador é utilizado para simular partes dos processos de registro e reconstrução.

Na holografia dinâmica existem três técnicas principais:

- **Holografia Eletrônica**: um computador é utilizado para simular o processo de registro das franjas de interferência no holograma à partir de uma descrição textual do objeto. Esta técnica requer bastante esforço computacional.
- **Escaneamento Volumétrico**: consiste na utilização de raios laser para projetar os pontos da superfície do objeto sobre uma tela giratória. O problema nesta técnica é a limitação mecânica devido ao fato do elemento de projeção ter que ser movido por toda área de exibição. Além disto, não é possível obter oclusão.
- **Holoprojeção**: consiste da projeção de luz contendo a informação espacial do objeto 3D sobre uma rede de difração especial, chamada `tela holográfica`[23, 24]. A tela holográfica é um filme transparente com propriedades de difração e foco. Ela refrata cada comprimento de onda da luz incidente em uma direção diferente (de forma similar a um prisma), desenhando assim o objeto no espaço 3D.

Neste trabalho estamos interessados na holoprojeção. Esta técnica requer uma fonte de luz, um mecanismo de codificação da informação espacial do objeto utilizando a luz desta fonte, e a tela holográfica. Este sistema é chamado `holoprojetor`.

Existem três versões de holoprojetores implementadas até o momento. Estas versões, apresentadas nas próximas seções, representam um histórico de sucessivas simplificações no processo de holoprojeção. A diferença básica entre elas está no mecanismo de codificação da informação espacial dos objetos utilizando a luz da fonte.

A - Holoprojetor 1.0

O primeiro sistema de Holoprojeção, chamado Holoprojetor 1.0, foi desenvolvido por Diamand e Lunazzi [10]. As componentes desta primeira versão são uma fonte de luz branca, três espelhos conectados a motores de passo para oscilação nos eixos cartesianos x, y e z (D_x, D_y e D_z), uma rede de difração reflexiva, um conjunto de lentes objetivas e a tela holográfica. O Holoprojetor 1.0 pode ser visto na Figura 3.3.

O funcionamento do sistema é bastante minucioso e o alinhamento de todas as peças para o perfeito resultado chega a ser um problema crítico. Para entendermos como tal sistema pode apresentar uma holografia dinâmica, vamos percorrer o caminho da luz, saindo de sua fonte até chegar à tela holográfica.

O foco de luz proveniente da fonte incide sobre o espelho D_x, que sendo movido pelo motor de passo, que por sua vez é comandado por um computador, desvia o feixe de luz

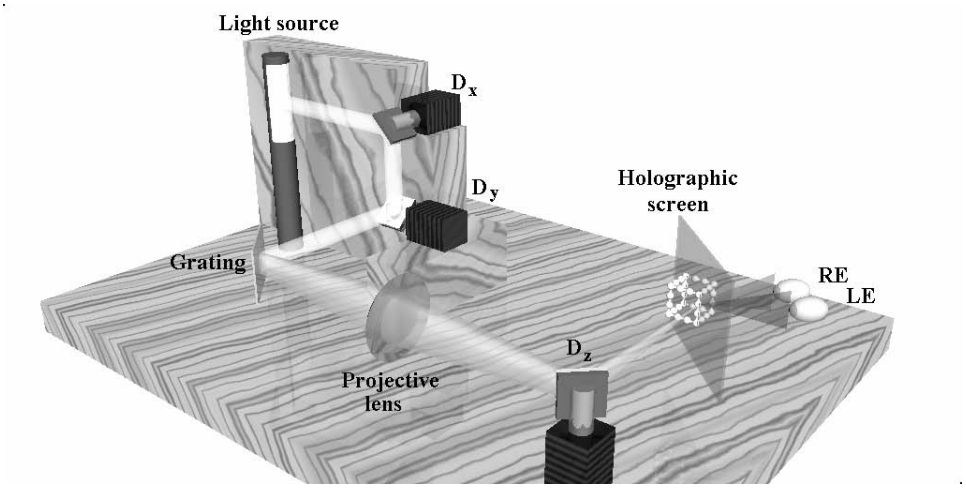


Figura 3.3: Holoprojetor 1.0

pelo eixo de coordenadas x. Logo após este espelho, encontra-se outro, o espelho D_y, que com as mesmas características, desloca o feixe de luz pelo eixo y. O feixe de luz incide sobre a rede de difração reflexiva. O ponto onde o feixe bate sobre a rede depende das coordenadas x e y tratadas pelos espelhos D_x e D_y. O mapeamento de todos os pontos destes dois espelhos forma um plano. Devido às características da rede de difração, o feixe de luz branca é dividido em dois raios de luz. O primeiro deles, no qual ficou a maior parte do brilho, não nos é útil. O segundo é um leque espectral de luz, sendo esta a parte da luz levada adiante. Estando a rede de difração na transversal, a largura deste leque dependerá do local de onde o feixe incide sobre a rede. O feixe será mais largo caso incida em um ponto mais distante com relação à lente objetiva, que é o próximo elemento do conjunto. Se o feixe incidir em um ponto da rede que esteja fisicamente mais próximo da lente, então terá menos espaço para sua abertura. A lente objetiva, recebendo o leque espectral, o converge formando um ponto de cruzamento do leque. O espelho D_z projeta esta convergência através da tela holografica, sendo que o ponto de cruzamento pode se formar antes ou depois de atingir a tela holográfica. Um observador posicionado em frente da tela holográfica perceberá que os pontos projetados são formados atrás da tela, na tela e na frente dela. O observador na Figura 3.3 está simbolizado por seus olhos direito (RE) e esquerdo (LE). Se o conjunto de espelhos D_x e D_y mapeia um plano, então o espelho D_z coloca este plano em diferentes posições através da tela holográfica (eixo z) formando um volume de exibição.

Por suas características o sistema descrito deve funcionar em completa escuridão, para que seja possível uma visualização adequada. Um recurso muito interessante que foi utilizado para a comprovação da real sensação 3D para o observador, foi a inclusão de uma fibra óptica devidamente iluminada e passando a uma distância determinada por

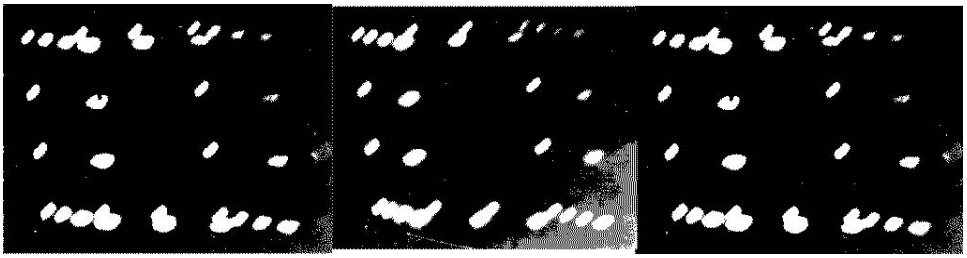


Figura 3.4: Resultado da visualização de uma imagem no Holoprojetor 1.0

trás e pela frente da tela, permitindo ao observador uma comparação com o espaço real.

Um dos resultados obtidos foi o desenho de um cubo com dezesseis pontos dispostos no espaço formando suas arestas e vértices. Tal resultado pode ser visto na Figura 3.4. Os problemas com esta versão são a lentidão do sistema e a perda de brilho. A lentidão é causada principalmente pelo movimentos dos espelhos. Incialmente os espelhos eram movidos a taxas bastante baixas, requerendo cerca de um segundo para a exibição dos dezesseis pontos que formam o cubo e um esforço por parte do observador para juntar toda a imagem exibida num único objeto. O motor de passo que movia os espelhos sofreu alterações no seu controle para aumentar a velocidade. Outros problemas surgiram, como a trepidação do espelho fazendo com que o ponto se desfocasse. Tal problema foi resolvido com a utilização de espelhos de acrílico.

No holoprojetor 1.0, a resolução da imagem exibida ainda é pequena, visto que o objeto em exibição é formado pelo conjunto de 16 pontos mapeados no espaço 3D.

B - Holoprojetor 2.0

A segunda versão do Holoprojetor foi na verdade uma melhora proposta por Bertini [3]. Suas modificações físicas incluem a retirada dos espelhos D_x e D_y, a substituição da fonte de luz por um projetor LCD e a substituição da rede de difração reflexiva por uma transparente.

Com a retirada dos espelhos D_x e D_y, que mapeavam os planos exibidos na tela holográfica, um objeto 3D tem que ser exibido nesta versão plano a plano. Isto é possível graças a utilização do projetor LCD. Entretanto o objeto deve ser previamente fatiado em planos consecutivos de forma similar ao fatiamento de um pão de forma. Este processo é feito usando um programa de *Ray Tracing* modificado para tal função. Cada um dos planos exibidos é posicionado pelo espelho D_z transversalmente através da tela holográfica. O restante do sistema permanece intacto como pode ser visto na Figura 3.5.

A Figura 3.6 mostra quatro planos de corte escolhidos para um objeto. O objeto é apresentado em par estéreo, para uma melhor visualização por parte do leitor, que pode utilizar as técnicas de olhar divergente ou convergente para vê-lo em 3D. A figura 3.7 mostra em par estéreo o resultado obtido com o Holoprojetor 2.0. Note que o objeto

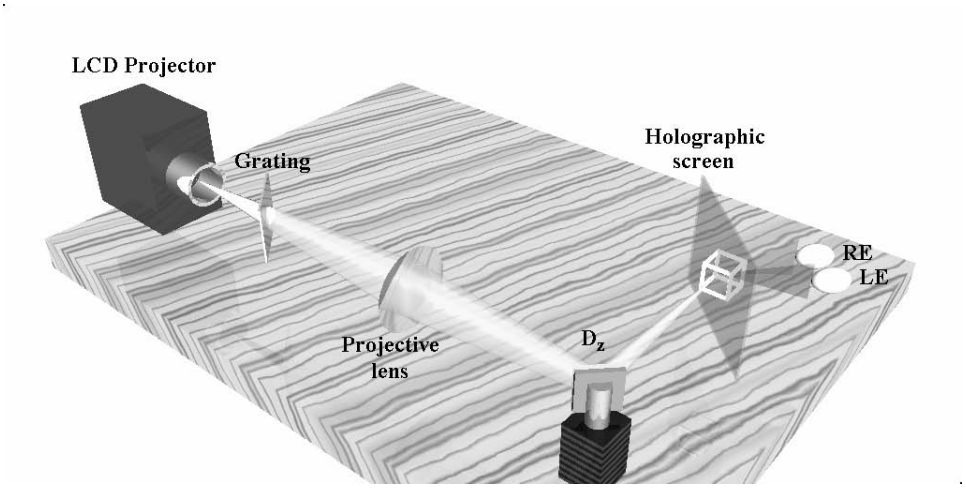


Figura 3.5: Holoprojetor 2.0

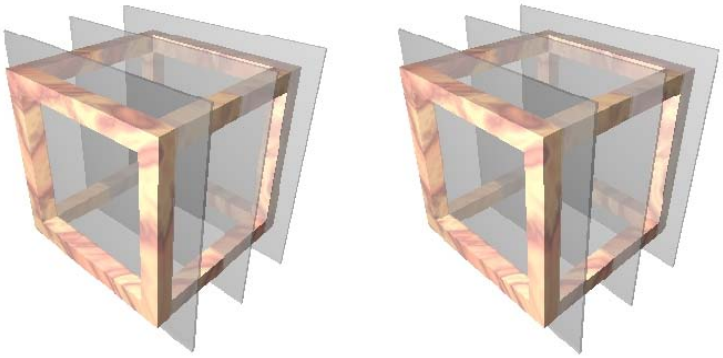


Figura 3.6: Planos de corte no processo de fatiamento

aparece ligeiramente espichado para as laterais, devido à sobreposição de fatias. Isto representa um problema, pois dependendo do ângulo de visão do observador, é possível enxergar os "buracos" entre cada fatia.

A função do espelho D_z é muito importante para o sistema. Apesar de operar a uma velocidade 6ms por passo, ou 180Hz, a exibição de um objeto usando quatro fatias reduz a taxa de exibição para 20Hz apenas. Requerendo uma seqüência repetitiva das fatias $\{(1,2,3),(4,3,2)\}$. Isto representa uma perda de resolução considerável. Além das perdas de brilho e foco que também podem ser observadas na Figura 3.7.

O Holoprojetor 2.0 sofreu ainda algumas melhorias que podem ser consideradas sua versão 2.1. O software foi ligeiramente modificado para exibir animações[8]. O número de fatias caiu de 4 para 3 atingindo uma taxa de exibição de 27Hz sem perda de qualidade

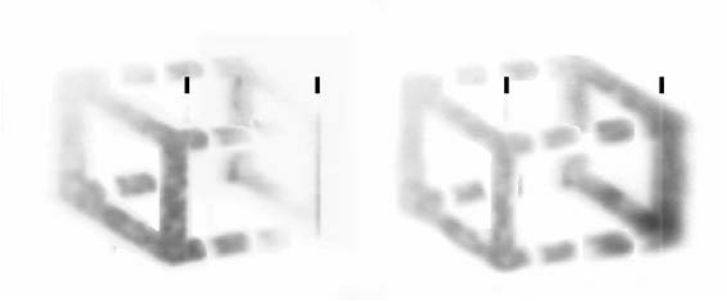


Figura 3.7: Resultado da visualização de uma imagem no Holoprojetor 2.0

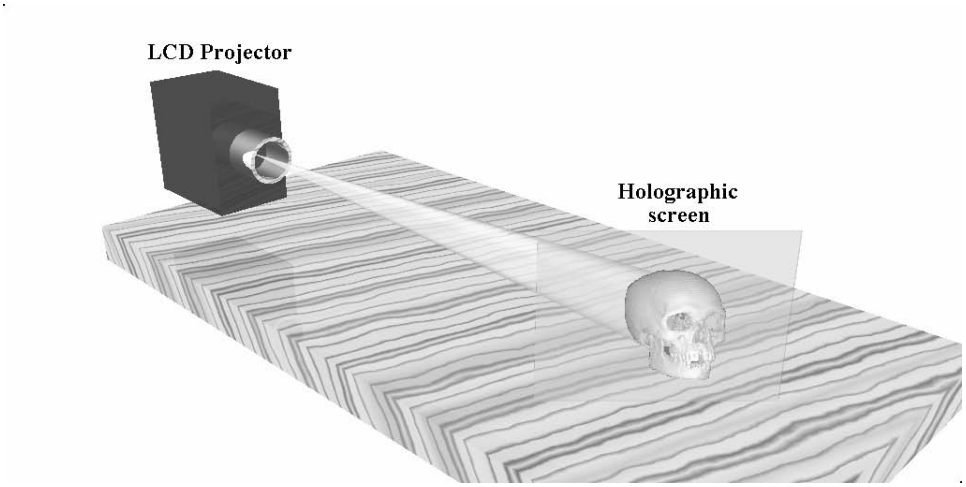


Figura 3.8: Holoprojetor 3.0

das imagens em exibição.

C - Holoprojetor 3.0

A terceira e atual versão do Holoprojetor recebeu melhorias bastante significativas. Aqui todos os aparatos intermediários entre o projetor e a tela holográfica foram removidos, tornando totalmente computacional o serviço de codificação cromática, que neste caso representa a profundidade da imagem 3D exibida. Esta versão do Holoprojetor foi proposta por da Fonseca em [7].

Como pode ser observado na Figura 3.8, a rede de difração, a lente objetiva e o espelho D_z foram removidos do conjunto, restando apenas o projetor LCD e a tela holográfica. As imagens 3D são projetadas diretamente sobre a tela holográfica, resolvendo assim dois grandes problemas: a resolução, que agora passa a ser praticamente a resolução do projetor (cerca de 800×600 pixels em média), e a melhoria de intensidade do brilho da imagem, que não sendo perdida ao longo do caminho da luz, faz com que esta versão possa trabalhar em uma tênue luz ambiente.

Sem a participação do espelho D_z, o objeto não precisa ser fatiado. Agora a codificação de profundidade é obtida cromaticamente usando a técnica de `codificação de estereogramas holográficos` apresentada a seguir.

3.4 Estereogramas Holográficos

A codificação de estereogramas holográficos se assemelha muito à técnica de anaglifos. A idéia aqui consiste em combinar diferentes vistas do objeto em paralaxe de movimento horizontal formando uma imagem 3D. Cada uma das vistas deve ser renderizada 2.5D em uma cor pura, ou melhor, em tons desta cor pura. Entre cada uma das vistas aconselha-se utilizar uma abertura máxima de 5 graus para obter paralaxe horizontal, caso seja utilizado mais do que isso, as imagens podem não ser integradas pelo observador. Para evitar distorções verticais entre as vistas, utiliza-se ainda rendering com projeção ortogonal. Como os atuais projetores LCD são baseados em três cores puras, vermelho, verde e azul (RGB), a codificação usa apenas três vistas do objeto em cada imagem 3D.

É importante notar que outras cores não são desejadas. Como o amarelo, por exemplo, que é na verdade representado por uma combinação do filtro vermelho e do filtro verde do sistema RGB do projetor. Essa combinação de cores é identificada pela tela holográfica que, neste caso, exibiria um ponto desfocado.

A exibição dos estereogramas holográficos usando o holoprojetor 3.0 mostra uma paralaxe de movimento horizontal discreta. Consideramos isto uma limitação do sistema, visto que o observador pode notar um pequeno salto de uma vista para a outra, quando ele se movimenta de um lado para o outro dentro do campo de observação. Por outro lado, já não existem mais os "buracos" observados na versão do holoprojetor 2.0.

3.5 Conclusão

O holoprojetor 3.0 apresenta uma alta qualidade na imagem de saída, em relação as versões anteriores. Nele tornou-se possível a exibição de cenas e animações 3D, num ambiente comum com pouca luz. Cabe ressaltar que quanto mais potente o projetor menos penunbra no ambiente é necessária. Sua simplicidade de montagem e uso também são pontos importantes a serem considerados. A utilização de estereogramas holográficos gerados por computador sobre este tipo de display 3D permite a criação de uma aplicação computacional para a visualização de qualquer estrutura 3D, contribuindo em áreas como medicina, química, matemática, etc.

Devido a este sistema ser auto-estereoscópico, ou seja, permite que o observador visualize a cena 3D sem o auxílio de dispositivos ou truques de visão, já é possível pensar

na substituição do monitor do computador por um "monitor holográfico", utilizando um holoprojetor e aplicações preparadas para utilizar estereogramas holográficos.

Baseados neste fato, desenvolvemos uma aplicação chamada HoloView, para visualização e estudo de imagens médicas 3D, que trabalha com estereogramas holográficos gerados pela técnica de shell rendering. Esta aplicação, que pode ser considerada como a primeira aplicação para o "monitor holográfico", será apresentada a seguir.

Capítulo 4

HoloView

4.1 Introdução

Este capítulo descreve em detalhes o sistema HoloView, para visualização 3D de imagens médicas, e as soluções adotadas para as dificuldades mencionadas no Capítulo 1.

O sistema HoloView consiste da seguinte seqüência de operações ilustradas na Figura 4.1. Dada uma cena 3D, a primeira etapa reúne as opções de pré-processamento que visam armazenar os dados para posterior visualização em tempo real. Podemos definir como visualização em tempo real para esta aplicação o fato do usuário interagir com a interface, girando a cena 3D, e o resultado é visto simultaneamente. Os estereogramas holográficos são codificados a partir dos dados armazenados e de parâmetros de visualização escolhidos pelo usuário na interface homem-computador (*loop* da Figura 4.1). Para cada variação nos parâmetros, uma nova imagem 3D (i.e. estereograma holográfico) é gerada e visualizada em tempo real no holoprojetor 3.0.

4.2 Pré-processamento

Lembrando a nossa discussão da Seção 2.3, onde uma cena 3D pode chegar a $60M$ voxels, o tempo para visualização vai depender muito de uma estrutura de dados eficiente para acessar os voxels de interesse rapidamente. Por isso adotamos o método *shell rendering* para a codificação dos estereogramas holográficos. Sua vantagem é armazenar apenas as informações sobre os voxels de interesse em uma estrutura especial denominada *shell*. Esta seção descreve as operações de pré-processamento adotadas para armazenar os dados nesta estrutura.

A etapa de pré-processamento consiste de três operações: interpolação, segmentação, e geração da estrutura *shell*. No HoloView existem duas opções de visualização que

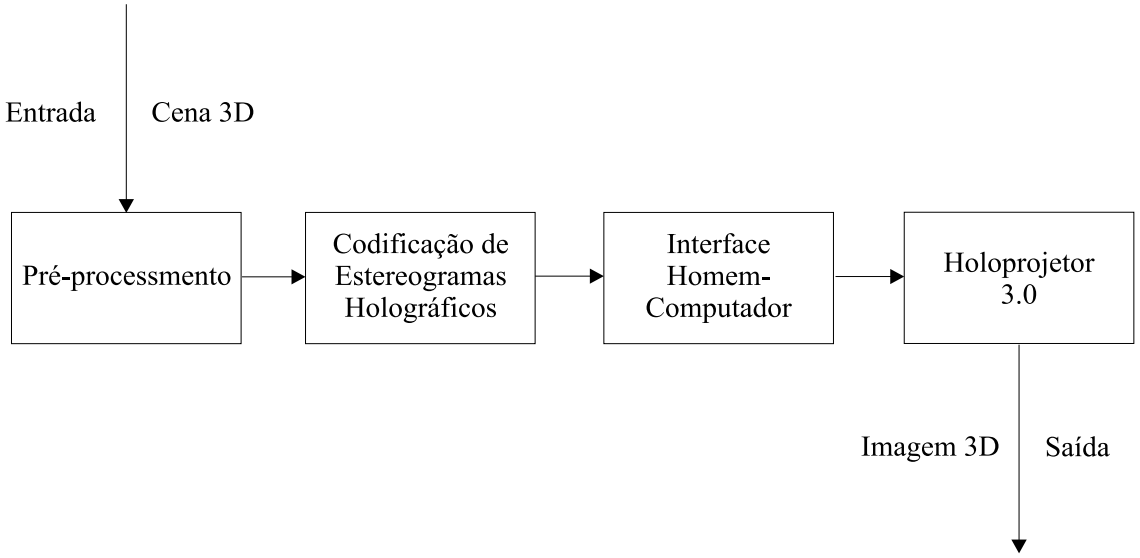


Figura 4.1: Diagrama de blocos das operações do HoloView.

combinam estas operações de forma diferente.

A primeira opção pode ser utilizada somente quando a cena original é uma cena numérica (em tons de cinza). A isotropia da cena é verificada automaticamente, e caso necessário, a cena é interpolada usando o método de interpolação linear descrito no Capítulo 2. A segmentação dos voxels a serem visualizados é feita *on-the-fly* por limiarização, conforme descrito no Capítulo 2, durante a etapa de codificação de estereogramas holográficos. Portanto, neste caso, todos os dados de visualização associados aos voxels com valor numérico diferente de zero devem ser armazenados na estrutura *shell*. Este tipo de *shell* é denominado *body shell*. Note que este tipo de *shell* poderia ser utilizado para rendering de volumes. No entanto, a técnica utilizada neste trabalho é a de rendering de superfícies.

A segunda opção aceita como entrada tanto uma cena binária, resultante de algum processo externo de segmentação, como uma cena numérica. No caso da cena numérica, a segmentação por limiarização é aplicada para selecionar os voxels da superfície do objeto a ser visualizado. Em ambos os casos, portanto, temos uma cena binária cuja isotropia é verificada automaticamente, e caso necessário, a cena binária é interpolada usando o método de interpolação baseada na forma, descrito no Capítulo 2. Neste método, nós adotamos o uso de um algoritmo rápido para transformada de distância Euclideana exata que é apresentado em detalhes na Seção 4.2.1. Este tipo de shell é denominado *surface shell*

Após a interpolação, os dados de visualização devem ser extraídos e armazenados na estrutura *shell*. Esta operação é descrita em detalhes na Seção 4.2.2.

4.2.1 Transformada de Distância Euclideana

Conforme discutido no Capítulo 2 sobre a interpolação baseada na forma, a transformada de distância 2D deve ser calculada dentro e fora do objeto em cada imagem da cena 3D binária. A interpolação linear aplicada aos valores de distância deve então considerar o sinal negativo para os valores de pixels fora do objeto e positivo para os pixels de objeto. Os métodos tradicionais aplicam a transformada de distância duas vezes, uma para os pixels de objeto e outra vez para os pixels de fundo. Uma contribuição neste trabalho é a interpolação baseada na forma através do cálculo rápido da transformada de distância, simultaneamente dentro e fora do objeto, e usando a métrica Euclideana no lugar da tradicional métrica de Chamfer [25].

A Figura 4.2 ilustra a transformada de distância Euclideana em uma imagem binária, onde os valores de distância associados a cada pixel são os valores quadráticos e o valor zero é associado a todos os pixels de borda. Nota-se que a raiz quadrada deste resultado e o acréscimo do sinal positivo/negativo para pixels de objeto/fundo é direto e deve ser calculado durante a etapa seguinte de interpolação linear. Portanto, o objetivo desta seção é mostrar apenas como se chega ao resultado da figura.

8	5	2	1	1	1	1	1	1	2	5
5	2	1	0	0	0	0	0	0	1	4
4	1	0	0	1	1	1	1	0	1	4
2	1	0	1	2	4	4	1	0	1	2
1	0	0	1	4	8	4	1	0	0	1
1	0	1	2	5	10	5	2	1	0	1
1	0	1	4	8	8	5	4	1	0	1
1	0	1	4	5	5	2	1	1	0	1
1	0	1	1	2	4	1	0	0	0	1
1	0	0	0	1	1	1	0	1	1	2
2	1	1	0	0	0	0	0	1	4	5
5	4	2	1	1	1	1	1	2	5	10

Figura 4.2: Transformada de distância Euclideana (valores quadráticos).

Neste trabalho adotamos o método sugerido em [12] para cálculo da transformada de distância Euclideana. Este método transforma o problema em um problema de floresta de caminhos mínimos. Nesta seção, vamos apenas simplificar sua explicação apresentando direto o algoritmo.

A - ALGORITMO TRANSFDIST

Entrada: Uma imagem binária I com dimensões $W \times H$ pixels;

Saída: Uma imagem numérica com os valores quadráticos da transformada de distância Euclideana (e.g. Figura 4.2)

Estruturas Auxiliares: Duas matrizes de deslocamentos $d_x(p)$ e $d_y(p)$, de dimensões $W \times H$, com o valor positivo do deslocamento em x e y, respectivamente, entre o pixel $p \in I$ e o pixel de borda em I mais próximo de p; uma fila hierárquica Q que armazena e remove os pixels de I na ordem crescente de hierárquia; uma matriz $s(p)$, de dimensões $W \times H$, que associa três situações possíveis para um pixel $p \in I$ com relação à fila Q: *inicial* - p nunca foi inserido em Q, *inserido* - p foi inserido em Q e *removido* - p foi removido de Q; e uma variável auxiliar *tmp* para o cálculo das distâncias;

início

1. *para* cada pixel $p \in I$ *faça*;
   (a) *se* p pertence a uma borda de objeto em I *então*;
      i. faça $d_x(p) \leftarrow 0$ e $d_y(p) \leftarrow 0$;
      ii. insira p em Q com hierarquia 0 e faça $s(p) \leftarrow inserido$;
   se não;
      i. faça $d_x(p) \leftarrow W$, $d_y(p) \leftarrow H$, e $s(p) \leftarrow inicial$;
2. *enquanto* Q não estiver vazia *faça*;
   (a) remova o pixel p de menor hierarquia em Q e faça $s(p) \leftarrow removido$;
   (b) *para* cada pixel q vizinho de p tal que $s(q) \neq removido$ *faça*;
      i. calcule $tmp \leftarrow (d_x(p) + |x_p - x_q|)^2 + (d_y(p) + |y_p - y_q|)^2$, onde (x_p, y_p) e (x_q, y_q) são as coordenadas (x, y) dos pixels p e q em I, respectivamente;
      ii. *se* $tmp < d_x^2(q) + d_y^2(q)$ *então*;
         A. calcule $d_x(q) \leftarrow d_x(p) + |x_p - x_q|$ e $d_y(q) \leftarrow d_y(p) + |y_p - y_q|$;
         B. *se* $s(q) \neq inserido$ *então*;
            insira q em Q com hierarquia *tmp* e faça $s(q) \leftarrow inserido$;
            se não;
            atualize a posição de q em Q com hierarquia *tmp*;

fim se;

fim enquanto;

fim

Existem alguns comentários com relação ao algoritmo acima. O primeiro diz respeito à corretude do algoritmo e a vizinhança dos pixels que é utilizada nos itens 1(a) e 2(b) [12]. Na maioria dos casos, a vizinhança-8 é suficiente para garantir o resultado correto, que requer simplesmente que todo pixel de borda tenha acesso a todos os pixels da imagem mais próximos dele. Esta premissa é necessária em qualquer algoritmo de transformada de distância. Mas em [6] está provado que dependendo das dimensões da imagem, vizinhanças maiores podem ser requeridas para garantir o sucesso da transformada de distância Euclideana. Este resultado é facilmente incorporado no algoritmo acima, bastando apenas trabalhar com uma vizinhança maior que 8. Um outro aspecto é o conceito de fila hierárquica discutido a seguir.

B - FILA HIERÁRQUICA

Uma fila hierárquica é uma estrutura de dados composta por N filas FIFO (*first-in-first-out*) com hierarquias $0, 1, \ldots, N-1$. Isto significa que um pixel com hierarquia h, $0 \leq h \leq N-1$, é sempre inserido no final da FIFO de hierarquia h e que, entre todos os pixels em uma FIFO de hierarquia h, o primeiro a ser removido é sempre aquele que está no início da FIFO. Os pixels de I são sempre armazenados na fila hierárquica Q do algoritmo TRANSFDIST na ordem crescente de hierarquia e o pixel removido é sempre o de menor hierarquia. No item 2(b)iiB, porém, a atualização da posição de um pixel em Q requer uma remoção do pixel independente da hierarquia e posição na FIFO, seguida de uma inserção no final da FIFO que representa o novo valor *tmp* de hierarquia.

Existem várias formas de implementar uma fila hierárquica. A nossa implementação é baseada na fila circular sugerida por Dial para o Algoritmo de Dijkstra [1]. Esta implementação tem a vantagem de manter a fila em tempo linear sem gastar muita memória, o que significa que o algoritmo TRANSFDIST calcula a transformada de distância Euclideana exata em tempo linear.

A fila hierarquica Q consiste de duas estruturas de dados: um vetor circular C e uma matriz A de ponteiros (ver Figuras 4.3 e 4.4). O vetor circular C possui N FIFOS implementadas em A como listas duplamente ligadas. Onde i_0 aponta sempre para a FIFO de menor hierarquia. A matriz A de ponteiros tem as mesmas dimensões $W \times H$ da imagem I e é pré-alocada para representar todas as possíveis FIFOS de C. Uma das vantagens deste algoritmo é que o tamanho N do vetor C não precisa ser igual ao maior valor *tmp* possível de hierarquia, calculado no item 2(b)i do algoritmo. Este valor

cresce com o quadrado da distância Euclideana e, portanto, pode chegar a valores muito altos. Na verdade, de acordo com Dial [1], a corretude do algoritmo requer que N seja no mínimo maior que o maior peso possível que pode ser atribuído a uma aresta no grafo. No nosso caso, o grafo está implícito na imagem, onde uma aresta é uma relação entre um par de pixels vizinhos. O maior peso de uma aresta é o maior valor possível para a diferença entre as distâncias Euclideanas quadráticas $[d_x^2(q) + d_y^2(q)] - [d_x^2(p) + d_y^2(p)]$, onde p é um pixel que foi removido de Q e q é um vizinho de p em I. Note que, para vizinhos 8, no pior caso $d_x(q) = W - 1$, $d_y(q) = H - 1$, $d_x(p) = W - 2$, $d_y(p) = H - 2$ e, portanto, $[d_x^2(q) + d_y^2(q)] - [d_x^2(p) + d_y^2(p)] = 2(W + H - 3)$. Como este valor tem que ser menor que N, escolhemos $N = 2(W + H - 3) + 1$. Isto significa que os pixels não precisam ser inseridos em Q com hierarquia tmp, mas sim, com hierarquia $mod(tmp, N)$, onde $mod(tmp, N)$ é o resto da divisão inteira de tmp por N.

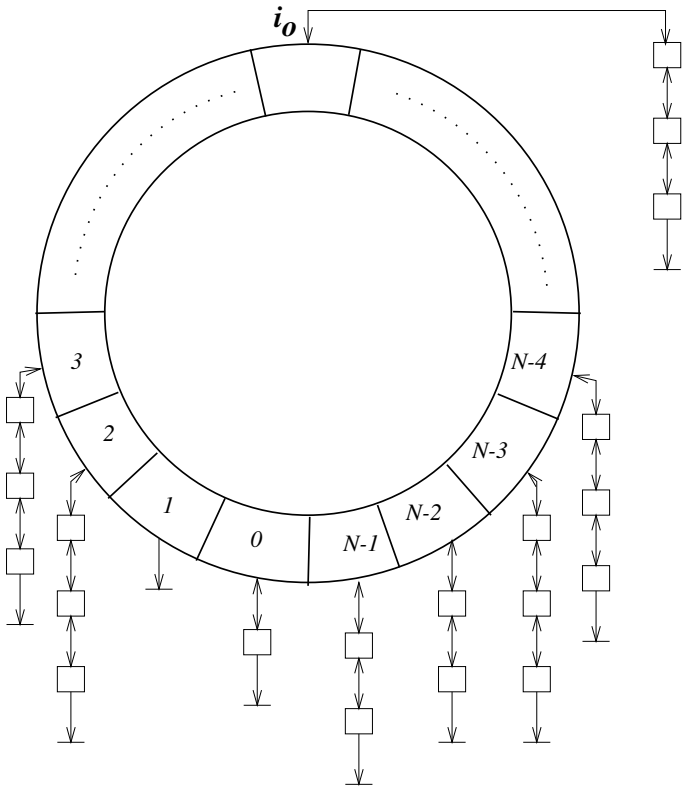


Figura 4.3: Vetor circular C usado na implementação da fila hierárquica Q. O índice i_0 aponta para a lista duplamente ligada que contém o pixel atual de menor hierarquia durante o processamento.

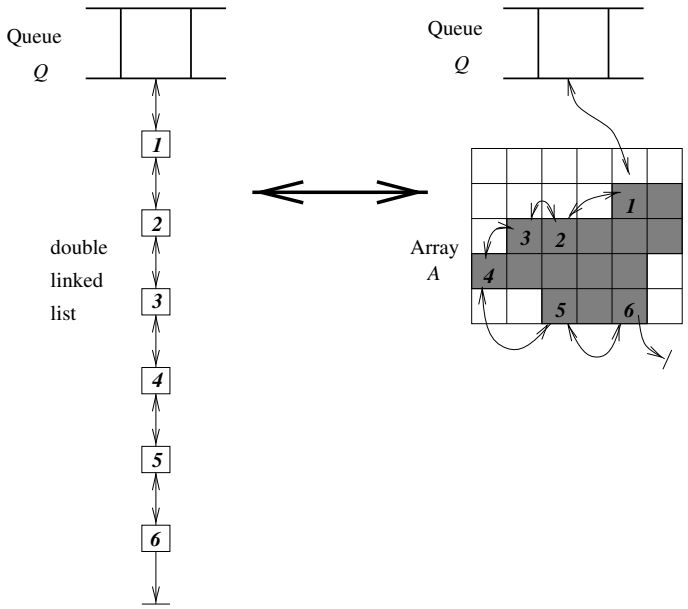


Figura 4.4: Matriz de ponteiros A usada para implementar as listas duplamente ligadas da fila circular C.

4.2.2 Estrutura *Shell*

A estrutura de dados *shell* é composta por uma lista indexada D e uma matriz de ponteiros P. Os voxels da cena 3D (ver Figura 2.2) são acessados coluna por coluna, linha por linha, imagem por imagem, iniciando na linha 0 ($y = 0$), coluna 0 ($x = 0$) e imagem 0 ($z = 0$). As informações necessárias para rendering dos voxels de interesse em cada uma das opções de visualização do HoloView são armazenadas nessa ordem na lista D. A matriz de ponteiros P tem dimensões $N \times H$, onde H é o número de linhas em uma imagem da cena 3D (eixo y da Figura 2.2) e N é o número de imagens (eixo z da Figura 2.2). Cada elemento (y, z) de P é um ponteiro para o primeiro elemento do *shell* em D que pertence à linha y da imagem z da cena 3D.

Na primeira opção de visualização, temos que os voxels de interesse pertencem ao *body shell*. Uma outra opção implementada neste trabalho é o uso de uma limiarização extra aplicada à função gradiente 3D dos voxels do *body shell*. Nossa intuição parte do princípio que os voxels de interface entre tecidos, onde o gradiente é mais alto, são os voxels de interesse. Com isso conseguimos uma boa redução das informações a serem armazenadas. Portanto, como a segmentação por limiarização ocorre durante o rendering, a lista D deve armazenar a posição x (coluna na cena 3D), o valor numérico v na cena 3D e o vetor gradiente $\vec{N}$ calculado para cada voxel selecionado, conforme a Equação 2.7. Este vetor representa uma aproximação do vetor normal à superfície do tecido mais denso que corta o voxel.

Na segunda opção de visualização, temos que os voxels de interesse pertencem ao objeto selecionado por limiarização ou qualquer outro processo externo de segmentação. Entre estes voxels, são selecionadas para armazenamento às informações relativas apenas aos voxels de superfície, formando uma casca de espessura 1 voxel. Isto representa uma economia de memória significativa, comparada à opção anterior, como veremos em alguns exemplos do capítulo 5. Note que, neste caso, porém, a lista D só precisa armazenar a posição x e o vetor $\vec{N}$ normal à superfície do objeto. Este vetor pode ser calculado na cena cinza interpolada ou a partir da própria cena binária da seguinte forma. Um filtro gaussiano 3D é aplicado à cena binária gerando uma cena numérica, e em seguida, os vetores normais são calculados na cena numérica usando a Equação 2.7.

O vetor normal $\vec{N} = (N_x, N_y, N_z)$ ocupa cerca de 50% do espaço de memória no *shell*. Para diminuir esta necessidade de armazenamento, foi implementada uma tabela fixa de valores das normais com 24.576 entradas. Consideramos para a criação desta tabela que cada voxel tem suas faces subdivididas em 64×64 pequenos quadrados e que o vetor normal discretizado atravessa o centro deles e estes valores são armazenados na tabela. Portanto, o vetor normal calculado é discretizado e a única informação relativa ao vetor que é armazenada no *shell* é o índice da tabela de normais.

Para finalizar, a Figura 4.5 mostra um exemplo 2D da estrutura de dados *shell*. Neste caso, P tem apenas as informações referentes as linhas y. Na parte superior da figura vemos a suposta cena 2D com vinte e cinco pixels. Os pixels mais escuros correspondem aos pixels que fazem parte do *shell* (objeto de interesse). Na parte inferior vemos a estrutura *shell* correspondente. Note que na estrutra shell original é necessário armazenar em P apenas os ponteiros para os elementos iniciais de cada linha em D. Nesta implementação, para aumentar a performance, utilizamos dois ponteiros em cada elemento de P, um apontadando para o primeiro elemento de cada linha em D e outro apontando para o último elemento em D da mesma linha. A lista D foi implementada como um vetor para facilitar o acesso seqüencial seja em ordem crescente ou decrescente do eixo x. A Seção 4.3.2 explica em detalhes a forma correta de acesso aos eixos para a cada direção de varredura FTB. A estrutura *shell* é sempre acessada na ordem z, y e x dos eixos, independente da forma de acesso crescente ou decrescente das coordenadas dos eixos. Uma outra estrutura *taboct* foi criada para armazenar as oito possíveis combinações da forma de acesso aos eixos referente à direção de varredura FTB.

4.3 Codificação de Estereogramas Holográficos

A codificação de estereogramas holográficos requer para cada ponto de vista do observador três renderings 2.5D com pequena paralaxe de movimento horizontal. Isto requer, inicialmente, a definição de um ambiente de visualização para simular a interação entre cena e

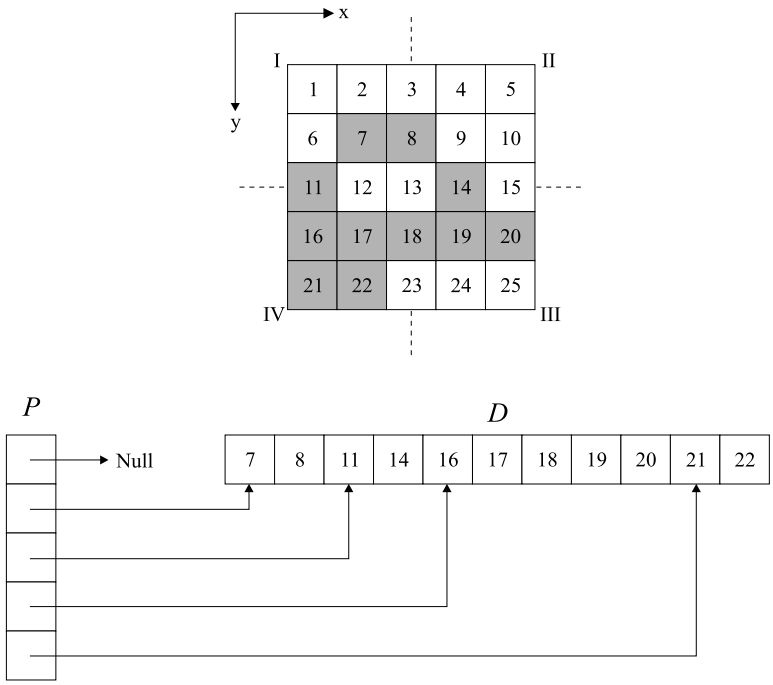


Figura 4.5: Exemplo de estrutura do shell

observador. Os três renderings 2.5D devem ser realizados e combinados rapidamente em uma única imagem RGB de modo que o observador não perca a noção de interatividade. Esta etapa requer uma modificação no algoritmo de *shell rendering* original.

4.3.1 Ambiente de Visualização

Definir um ambiente de visualização significa definir a posição de cada elemento, cena, observador, fontes de luz, e plano de visualização. No HoloView, o observador e a única fonte de luz ficam no $z = -\infty$ (Figura 4.6). A cena 3D deve ser centrada na origem do sistema x, y, z, com as imagens tomográficas paralelas ao plano x, y. Note que isto implica em uma translação $(-W/2, -H/2, -N/2)$ dos voxels da cena original, onde W e H são largura e altura de cada imagem, e N é o número de imagens da cena 3D. Assim, qualquer ponto de vista da cena pode ser obtido rotacionando-a de um ângulo α em torno de x (*tilt*) e de um ângulo β em torno de y (*spin*). O plano de visualização é posicionado em $z = D/2$, onde D é a maior diagonal da cena 3D. Isto garante que independente da rotação realizada, a cena sempre estará na frente do plano. Para que, independente da rotação, todos os voxels sejam projetados na imagem sobre o plano de visualização, nós escolhemos uma imagem de dimensões $D \times D$, com origem em $x = -D/2$ e $y = -D/2$. Note que isto implica em outra translação de $(D/2, D/2, 0)$ dos voxels que estão sendo projetados sobre o plano. Portanto, podemos resumir o mapeamento de um voxel $v(x, y, z)$ em um

pixel $p(u, v)$ com as seguintes transformações em coordenadas homogêneas:

$$\begin{bmatrix} u \\ v \\ d(u,v) \\ 1 \end{bmatrix} = T_D \times R_\beta \times R_\alpha \times T_O \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{4.1}$$

onde
$[x, y, z, 1]$ são as coordenadas homogêneas do voxel $v(x, y, z)$;
T_O é a matriz de translação para a origem do sistema;
R_α é a matriz de rotação α em torno do eixo x;
R_β é a matriz de rotação β em torno do eixo y;
T_D é a matriz de translação e projeção do voxel na imagem do plano de visualização.
$[u, v, d(u, v), 1]$ são as coordenadas homogêneas do pixel $p(u, v)$ na imagem e $d(u, v)$ é a distância do voxel ao plano de visualização utilizada para calcular a tonalização por profundidade na Equação 2.4;

$$\begin{bmatrix} u \\ v \\ d(u,v) \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & D/2 \\ 0 & 1 & 0 & D/2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} cos\beta & 0 & -sen\beta & 0 \\ 0 & 1 & 0 & 0 \\ sen\beta & 0 & cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha & -sen\alpha & 0 \\ 0 & sen\alpha & cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & -W/2 \\ 0 & 1 & 0 & -H/2 \\ 0 & 0 & 1 & -N/2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}; \tag{4.2}$$

Note que, cortes na cena 3D são feitos no HoloView aplicando um simples limiar em $d(u, v)$.

Definido o ambiente de visualização e como os voxels são mapeados nos pixels da imagem a ser gerada, a idéia agora é definir a estrutura de dados que deve ser usada no lugar da cena 3D da Figura 4.6.

4.3.2 Shell Rendering

O *shell rendering* consiste de duas operações: projeção e tonalização. A tonalização é feita conforme descrito no Capítulo 2. Esta seção aborda apenas os detalhes de projeção.

Para efetuar a projeção dos dados contidos no shell primeiro é necessário identificar qual dos oito possíveis octantes da cena 3D está posicionado na direção da visão do

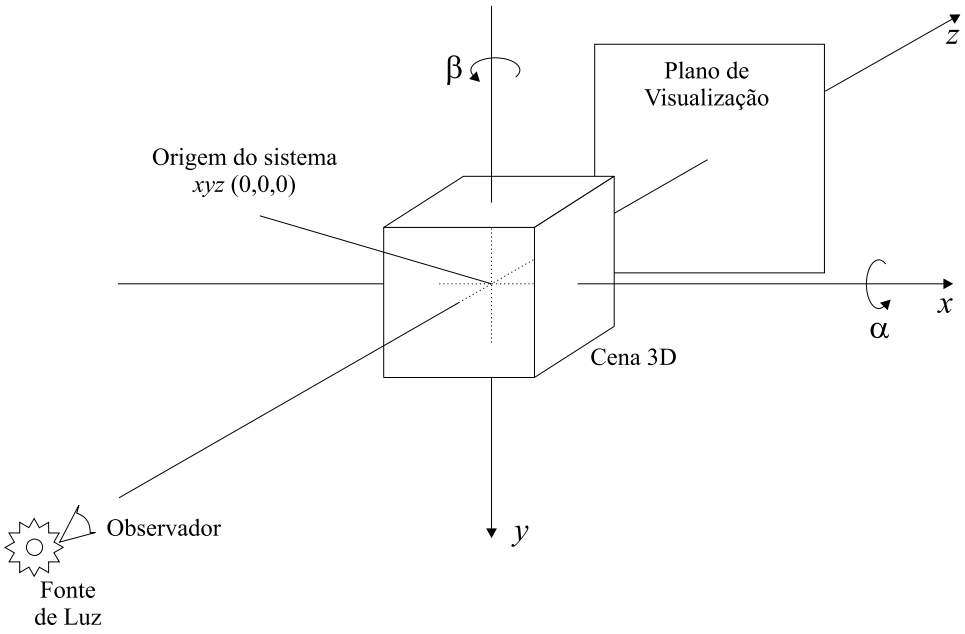


Figura 4.6: Ambiente de visualização

observador, conforme pode ser visto na Figura 2.6. Para cada um dos octantes existe uma direção de varredura FTB do shell, garantindo sempre que o voxel mais próximo do observador será o primeiro e o mais distante será o último a ser projetado sobre o mesmo pixel do plano de visualização. O sentido de indexação dos voxels para cada um dos casos pode ser visto na Tabela 2.1. A notação utilizada $x^- \rightarrow x^+$ indica que os valores do eixo x devem ser percorridos da menor para a maior coordenada. É importante notar que não existe nenhuma restrição quanto à ordem de precedência entre os eixos x, y e z e, por simplicidade, no HoloView o acesso FTB foi implementado fatia por fatia (z), linha por linha (y) e coluna por coluna (x).

Como exemplo de acesso FTB, o shell mostrado na Figura 4.5 representa o caso 2D, onde as direções de visualização podem ser definidas com base apenas no ângulo α. Para cada intervalo de 90 graus de α existe uma ordem de acesso aos pixels conforme mostrado na lista abaixo. No caso 3D, com dois ângulos para a rotação, existem oito combinações entre dos intervalos de 90 graus de α e β.

1. Para $0^0 \leq \alpha \leq 90^0$

   FTB: 5, 4, 3, 2, 1, 10, 9, 8, 7, 6, 15, 14, 13, 12, 11, 20, 19, 18, 17, 16, 25, 24, 23, 22, 21.

2. Para $90^0 < \alpha \leq 180^0$

FTB: 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1.

3. Para $180^0 < \alpha \leq 270^0$

   FTB: 21, 22, 23, 24, 25, 16, 17, 18, 19, 20, 11, 12, 13, 14, 15, 6, 7, 8, 9, 10, 1, 2, 3, 4, 5.

4. Para $270^0 < \alpha < 360^0$

   FTB: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25.

A cada voxel visitado no shell pela ordem FTB, é aplicado nele as matrizes de rotação conforme mostrado na Equação 4.2. Se o pixel $p(u, v)$ no plano de projeção ainda não estiver com algum valor de projeção atribuído, é calculada a tonalização de $p(u, v)$ de acordo com a Equação 2.2. Cabe lembrar, que no HoloView, apenas o rendering de superfícies foi implementado, por tanto o pixel projetado é considerado completamente opaco, sendo que qualquer outro voxel projetado na mesma coordenada (u, v) não contribui em nada para a tonalização de $p(u, v)$. O fato de não ser necessário tonalizar mais do que uma vez um pixel no plano de projeção, conseguido através do acesso FTB da estrutura Shell, é uma das grandes vantagens desse método em relação aos outros métodos de rendering, que o torna um dos algoritmos mais rápidos.

A tonalização garante ao pixel $p(u, v)$ um tom de cinza, que faz sentido em um rendering 2.5D, mas para a geração dos estereogramas holográficos, precisamos considerar tons de vermelho (R), verde (G) e azul (B) e combiná-los numa mesma imagem. O rendering de cada uma das camadas de tons é feito separadamente com uma pequena variação no ângulo β para a obtenção da paralaxe horizontal. O valor do pixel $p(u, v)$ em cada um dos renderings está no intervalo $[0, 255]$, de forma que o primeiro rendering pode ser considerado como tons de vermelho no intervalo $[0, 255]$, o segundo rendering como tons de verde e o terceiro como tons de azul, gerando um pixel final com cores reais de 24 bits.

4.3.3 Algoritmo

Esta seção apresenta o algoritmo de shell rendering modificado para incluir a projeção dos estereogramas holográficos. As variáveis e estruturas auxiliares relevantes para o algoritmo são:

α e β: Tilt e spin respectivamente, para a rotação da cena, fornecidos pelo usuário;
taboct: tabela de indexação FTB;

x, y e z: coordenadas do voxel conseguidas pelo acesso matriz B e lista P de acordo com *taboct*;
N_x, N_y e N_z: Normal discretizada do vetor de normais;
$p1(u,v)$, $p2(u,v)$ e $p3(u,v)$: Matrizes de pixels usadas para armazenar os renderings R, G e B;
$p(u,v)$: Matriz de pixels usada para armazenar a composição final do rendering RGB;

início

1. Calcule a tabela de octante *taboct* para acesso FTB através de α e β;
2. *para* cada z na ordem de acesso FTB faça;
   (a) *para* cada y na ordem de acesso FTB faça;
      i. *para* cada x na ordem de acesso FTB faça;
         A. calcule $(u, v, d(u,v)) \leftarrow T_D \times R_\beta \times R_\alpha \times T_O \times (x, y, z)$;
         B. *se* $p1(u,v) = 0$ então;
            calcule $(n_x, n_y, n_z) \leftarrow T_D \times R_\beta \times R_\alpha \times T_O \times (N_x, N_y, N_z)$;
            calcule θ entre a normal e a direção de visualização, usando $arccos(n_z)$;
            calcule $p1(u,v) \leftarrow T_{amb} + T_{dist}(u, v, d(u,v))(T_{dif}(u, v, \theta) + T_{esp}(u, v, \theta))$ (Equação 2.2);
            fim se;
         fim para;
      fim para;
   fim para;
3. *se* geração de estereogramas holográficos, *então*
   (a) repita passos 1 e 2 com β + *paralaxe horizontal* e armazenando o resultado em $p2(u,v)$ em vez de $p1(u,v)$;
   (b) repita passos 1 e 2 com β - *paralaxe horizontal* e armazenando o resultado em $p3(u,v)$ em vez de $p1(u,v)$;
   (c) *para* cada pixel p no plano de visaulização faça;
      i. gere imagem 24-bits $p(u,v)$ usando $p1(u,v)$, $p2(u,v)$ e $p3(u,v)$ como componentes RGB, respectivamente;
   fim para

se não;

4. *para* cada pixel p no plano de projeção faça;

   (a) calcule $p(u, v) \leftarrow p1(u, v)$;

   fim de se;

fim

4.4 Interface Homem-Computador

A interface do HoloView é bastante simplificada, visando atingir seu objetivo de ferramenta de visualização e manipulação de imagens médicas sem confundir o usuário. A etapa de pré-processamento necessita ser executada apenas uma vez para cada cena, quando são fornecidas as imagens que compõem a cena original, as proporções das medidas dos voxels e o tipo de shell a ser gerado. Nesta etapa há um esforço computacional muito grande para a geração do shell devido a discretização das normais. Com o shell armazenado, passamos para a etapa de geração de estereogramas holográficos que demanda a maior parte dos parâmetros da interface homem-computador.

Como pode ser visto na Figura 4.7, estão colocadas na tela principal as ferramentas mais utilizadas, sendo: `Tilt`, `Spin`, `Rendering size`, `Threshold`, `Horizontal parallax`, `Cut plane`, opção de `filtro de buracos`, opção `on/off line rendering`, opção de `tipo de rendering` e o `botão de rendering`.

A opção de Rendering size, reamostra a imagem de saída no intervalo $[256 \times 256, 1024 \times 1024]$ pixels. A opção de Threshold é utilizada para produzir o *threshold on-the-fly* com o shell preparado para rendering volumétrico. Horizontal parallax configura o ângulo de abertura entre as vistas para o caso 3D. É possível executar 255 cortes em diferentes posições no objeto, limiarizando $d(u, v)$ em 0, sendo os cortes estipulados em uma percentagem da profundidade da cena 3D sobre o eixo z, tomando-se como base a cena já rotacionada em α e β. O rendering pode ser on line ou off line, dependendo da velocidade da máquina que abriga a aplicação. Na opção de tipo de rendering é possivel escolher entre um rendering 2.5D, rendering 3D com vista estéreo para uso com o monitor e óculos vermelho-azul, rendering 3D para Holoprojetor 3.0 e rendering 2D para *maximum intensity projection*.

Outras opções como `animação` e exportação das imagens geradas também foram implementadas, permitindo a visualização de animações 2.5D e 3D, em monitor convencional ou holoprojetor 3.0 respectivamente e o salvamento de qualquer rendering gerado.

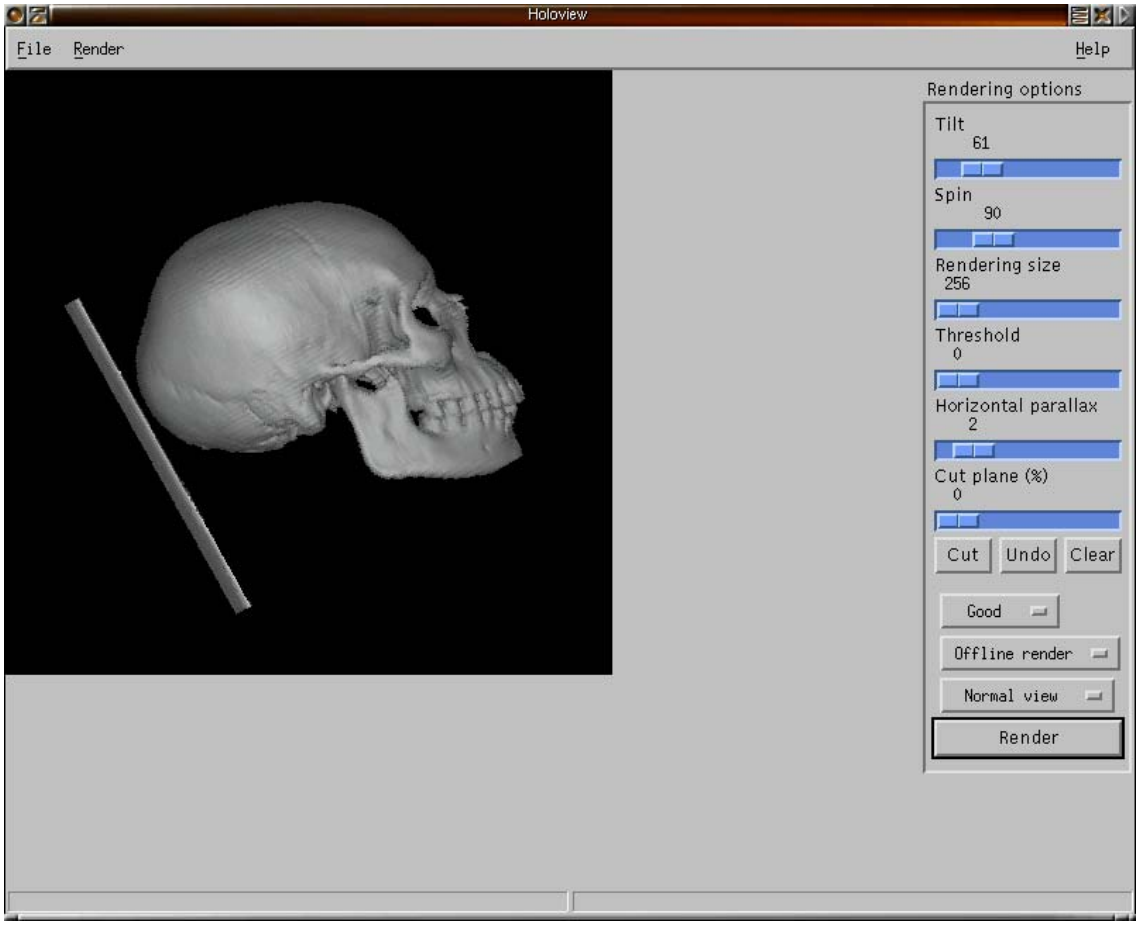


Figura 4.7: Tela principal da aplicação HoloView.

O HoloView foi implementado em linguagem C++ com a biblioteca gráfica wxWindows 2.0 para Motif. Esta plataforma de desenvolvimento permite uma grande portabilidade entre sistemas como Unix (a plataforma original de desenvolvimento foi o Linux, versão Slackware 7.0), Windows e Macintosh, com uma variedade grande de diferentes compiladores C++. O compilador utilizado no Linux foi o Egcs 2.91.66 (gcc). Uma versão do HoloView para plataforma Windows com ocompilador Cygnus B20 e a biblioteca wxWindows 2.16 está em construção. Para os sistemas Unix sem Motif, existe uma versão da biblioteca gráfica que utiliza o Gtk.

4.5 Conclusão

Neste Capítulo apresentamos o HoloView, uma ferramenta para visualização de estereogramas holográficos em computação de imagens médicas. A principal contribuição do HoloView é possibilitar a criação de um sistema de visualização realmente 3D com o Holoprojetor 3.0, em substituição ao monitor convencional. Em sua implementação optamos por utilizar a técnica de shell rendering, que atualmente pode ser considerada uma das mais rápidas entre as técnicas de rendering. Este fator é de grande importância para a geração dos estereogramas holográficos pois são necessários 3 renderings distintos, triplicando o tempo de processamento para cada imagem gerada.

A divisão da aplicação em duas etapas, a de pré-processamento e a de geração dos estereogramas permitiu a criação de uma ferramenta de fácil manipulação até mesmo por pessoas com muito pouco conhecimento na área de processamento de imagens. Por isto, parâmetros relacionados com as técnicas de interpolação linear e interpolação baseada na forma, por exemplo, são omitidos do usuário que não precisa saber de todos os detalhes necessários para a visualização. Outros sistemas se tornam de difícil utilização, porque não omitem estes detalhes.

Outras opções implementadas como cortes, animação, e tipo de rendering tornam o HoloView uma ferramenta interessante para a manipulação e o estudo de imagens médicas, mesmo sem a utilização do holoprojetor 3.0. A escolha de uma biblioteca gráfica altamente portável para a criação da interface criou uma aplicação praticamente independente de plataforma, visto que não é necessário nenhum hardware específico (exceto o Holoprojetor 3.0 para um sistema auto-estereoscópico) e que o sistema pode ser recompilado para uma grande número de plataformas.

Capítulo 5

Resultados

Este capítulo apresenta os resultados de experimentos realizados com respeito ao tempo de processamento e qualidade da imagem final do sistema Holoview, comprovando sua viabilidade para visualização e manipulação de imagens médicas.

5.1 Tempo de Processamento

O HoloView pode ser logicamente dividido em uma etapa de pré-processamento e uma etapa de rendering 3D. O pré-processamento é a etapa mais custosa, e pode levar alguns minutos, envolvendo as operações de interpolação, segmentação, cálculo das normais e geração do shell. Enquanto o rendering 3D envolve a geração e a projeção de estereogramas holográficos na tela holográfica e deve ocorrer em tempo real (i.e. em torno de 1 segundo).

Para mostrar a viabilidade do sistema HoloView com relação ao tempo de processamento, escolhemos testá-lo em uma plataforma de baixo custo e poder computacional. Portanto, os experimentos foram realizados em um Pentium MMX 166MHz com 48MBytes de memória RAM e placa de vídeo Trident 9680 com 2MBytes de memória.

Nesses experimentos foram utilizadas quatro diferentes cenas sendo elas: um crânio seco (C), região da garganta e vias respiratórias de um paciente (G), o joelho de um paciente (J) e parte da fíbula (F). As três primeiras cenas são provenientes de tomografia de Raios-X e a última de ressonância magnética. À partir destas cenas, montamos variações das cenas com características diferentes entre si, a saber:

- C_1: Cena em tons de cinza não interpolada (crânio);
- C_2: Cena binária não interpolada;
- C_3: Cena binária e cena em tons de cinza para cálculo da normal à superfície do voxel não interpoladas;

	Tempo	Voxels do shell	Redução
$SShC_1$	1min06s	305.176	98%
$BShC_1$	2min43s	2.320.589	86%
$SShC_2$	6min33s	286.510	98%
$SShC_3$	4min24s	286.510	98%
$SShJ_1$	30s	162.629	97%
$BShJ_1$	2min23s	2.399.920	64%

Tabela 5.1: Tabela de tempos de pré-processamento para a geração do shell incluindo interpolação.

- C_4: Cena interpolada previamente em tons de cinza;
- C_5: Cena binária interpolada previamente;
- C_6: Cena binária e cena em tons de cinza para cálculo da normal à superfície do voxel interpoladas previamente;
- G_1: Cena em tons de cinza previamente interpolada (garganta);
- J_1: Cena em tons de cinza não interpolada (joelho);
- F_1: Cena binária previamente interpolada (fíbula).

Para todas as cenas em tons de cinza pode-se gerar tanto *body shell* (BSh), que compreende todos os voxels com valor diferente de zero na cena, como *surface shell* (SSh), que contém somente os voxels da superfície do objeto. Entretanto, utilizando-se as cenas binárias como entrada, apenas o *surface shell* é aplicável. Podemos considerar como o pior caso para a geração do shell, independente do tipo da cena, quando esta deve ser interpolada automaticamente antes do processo de segmentação, e o melhor caso quando consideramos a cena já interpolada como entrada.

Foram geradas, à partir das cenas do crânio, oito cenas diferentes. Considerando que a cena interpolada correspondente tem 16.252.928 voxels, as cenas $SShC_1$, $BShC_1$, $SShC_2$ e $SShC_3$ apresentam uma redução de 98%, 86%, 98% e 98% quando são armazenadas no shell. Para as cenas $SShJ_1$ e $BShJ_1$, que provêm da cena do joelho, a redução é de 97% e 64%, sendo que a cena interpolada correspondente possui 6.684.672 voxels. Estes resultados, que representam os piores casos de geração do shell, juntos ao tempo de pré-processamento destas cenas são apresentados na Tabela 5.1.

As cenas do crânio apresentadas na Tabela 5.2 devem refletir o mesmo fator de redução das cenas do crânio da Tabela 5.1, uma vez que o número de voxels para as cenas é similar. No caso das cenas $SShG_1$ e $BShG_1$ da garganta, esta redução é de 97% e 26%, respectivamente, com relação aos 5.439.488 voxels da cena original. A cena da fíbula

	Tempo	Voxels do shell	Redução
$SShC_4$	36s	306.374	98%
$BShC_4$	2min14s	2.310.901	86%
$SShC_5$	3min21s	322.526	98%
$SShC_6$	39s	322.526	98%
$SShG_1$	15s	149.622	97%
$BShG_1$	4min49s	4.003.529	26%
$SShF_1$	5s	11.148	95%

Tabela 5.2: Tabela de tempos de pré-processamento para geração de shell com cenas já interpoladas.

$SShF_1$ apresentou fator de redução de 95% considerando os 224.352 da cena original. Os resultados apresentados nesta tabela representam os melhores casos de geração de shell, uma vez que a interpolação foi feita previamente fora do HoloView.

Os resultados comparativos entre o rendering de imagens 2.5D e o de estreogramas holográficos 3D são apresentados a seguir. Existem, no HoloView, 4 diferentes formas de rendering: (a) Rendering convencional 2.5D, denominado *Normal view*, (b) Rendering de estereograma holográfico para visualização com monitor de computador e óculos vermelho-azul, composto apenas por duas vistas (RB), *Stereo view*, (c) Rendering de estereogramas holográficos para Holoprojetor 3.0 de três vistas (RGB), *3D view* e (d) Maximum intensity projection 2D, *M.I.P.*. Devido ao aparecimento de pequenos buracos na imagem de saída causado por fatores já mensionados no Capítulo 4, foram implementados dois filtros de fechamento de buracos *Good* e *Perfect*. A opção *Normal* não apresenta nenhum tipo de filtro de fechamento de buracos. A Tabela 5.3 apresenta os tempos de rendering para os shells $SShC_1$, $BShC_1$, $SShC_2$ e $SShC_3$ em todas as combinações possíveis de renderings e filtros. Todos os tempos representam a média aritmética dos tempos de vários renderings com diferentes valores para tilt e spin e são apresentados em segundos. A Tabela 5.4 mostra os tempos de renderings para os shells $SShC_4$, $BShC_4$, $SShC_5$ e $SShC_6$, cujos valores são muito próximos aos valores mostrados na Tabela 5.3, visto que o número de voxels nos shells são similares.

Nos casos dos shells $SShG_1$ e $BShG_1$ os tempos de rendering (em segundos) para todas as combinações possíveis são mostrados na Tabela 5.5, e para os shells $SShJ_1$ e $BShJ_1$ os tempos de rendering são mostrados na Tabela 5.6. A Tabela 5.7 mostra os tempos de rendering para o shell $SShF_1$.

Podemos dizer que os tempos de pré-processamento e rendering apresentados nas Tabelas 5.3, 5.4, 5.5, 5.6 e 5.7 estão dentro do aceitável no caso de um Pentium 166MHz. Atualmente, porém, podemos considerar um Pentium II 400MHz como um computador de baixo custo e poder computacional. Neste caso, os tempos apresentados serão bem

	$SShC_1$			$BShC_1$		
	Normal	Good	Perfect	Normal	Good	Perfect
Normal view	0,832	0,903	1,010	2,698	2,782	2,876
Stereo view	1,665	1,828	2,008	5,403	5,571	5,762
3D view	2,496	2,773	3,040	8,108	8,359	8,644
M.I.P.	0,549	0,623	0,696	2,116	2,235	2,312
	$SShC_2$			$SShC_3$		
	Normal	Good	Perfect	Normal	Good	Perfect
Normal view	0,798	0,883	0,975	0,800	0,883	0,976
Stereo view	1,603	1,769	1,958	1,605	1,772	1,958
3D view	2,407	2,657	2,938	2,412	2,665	2,944
M.I.P.	-	-	-	-	-	-

Tabela 5.3: Tabela de tempos de rendering (em segundos) para os shells $SShC_1$, $BShC_1$, SSh_C2 e $SShC_3$.

	$SShC_4$			$BShC_4$		
	Normal	Good	Perfect	Normal	Good	Perfect
Normal view	0,845	0,942	1,020	2,717	2,783	2,883
Stereo view	1,688	1,835	2,020	5,406	5,572	5,764
3D view	2,519	2,774	3,040	8,108	8,365	8,657
M.I.P.	0,553	0,623	0,701	2,169	2,242	2,319
	$SShC_5$			$SShC_6$		
	Normal	Good	Perfect	Normal	Good	Perfect
Normal view	0,879	0,957	1,064	0,878	0,962	1,058
Stereo view	1,758	1,924	2,098	1,764	1,934	2,126
3D view	2,627	2,868	3,146	2,650	2,905	3,192
M.I.P.	-	-	-	-	-	-

Tabela 5.4: Tabela de tempos de rendering (em segundos) para os shells $SShC_4$, $BShC_4$, $SShC_5$ e $SShC_6$.

	$SShG_1$			$BShG_1$		
	Normal	Good	Perfect	Normal	Good	Perfect
Normal view	0,380	0,427	0,476	5,491	5,559	5,634
Stereo view	0,763	0,856	0,954	10,996	11,113	11,260
3D view	1,146	1,284	1,433	16,484	16,679	16,877
M.I.P.	0,257	0,295	0,335	4,742	4,801	4,849

Tabela 5.5: Tabela de tempos de rendering (em segundos) para os shells $SShG_1$ e $BShG_1$.

	$SShJ_1$			$BShJ_1$		
	Normal	Good	Perfect	Normal	Good	Perfect
Normal view	0,373	0,413	0,454	3,493	3,545	3,599
Stereo view	0,748	0,827	0,910	6,990	7,092	7,202
3D view	1,122	1,242	1,366	10,487	10,640	10,805
M.I.P.	0,274	0,306	0,339	3,022	3,065	3,109

Tabela 5.6: Tabela de tempos de rendering (em segundos) para os shells $SShJ_1$ e $BShJ_1$.

	$SShF_1$		
	Normal	Good	Perfect
Normal view	0,040	0,047	0,055
Stereo view	0,079	0,095	0,111
3D view	0,119	0,142	0,166
M.I.P.	-	-	-

Tabela 5.7: Tabela de tempos de rendering (em segundos) para o shell $SShF_1$.

menores justificando ainda mais a viabilidade do sistema HoloView.

5.2 Qualidade de Imagem

A Figura 5.1 mostra duas imagens 3D geradas pelo sistema HoloView. Estas imagens foram captadas com uma câmera fotográfica Canon EOS Rebel II, num ambiente com pouca luminosidade e iluminação uniforme. Nesta imagem, quando vista no Holoprojetor, pode-se observar cinco vistas diferentes do objeto. As três vistas RGB do estereograma holográfico, que são separadas em diferentes comprimentos de onda pela tela holográfica, e duas outras vistas intermediárias resultantes das combinações RG e GB. Movendo a cabeça de um lado para o outro em frente à tela, pode-se ver a troca das diferentes vistas da cena com um levíssimo salto entre elas, o que podemos caracterizar como uma paralaxe horizontal quase contínua. O tamanho dos saltos entre as vistas está diretamente relacionado com o ângulo de abertura entre os renderings R, G, e B para a codificação do estereograma holográfico. Durante os experimentos, encontramos algumas dificuldades relacionadas à fatores físicos da montagem do Holoprojetor que podem prejudicar a qualidade da holografia. Uma das dificuldades é a concentração do foco do projetor LCD, que varia conforme o equipamento. A distância da tela até o projetor e do observador até a tela também é variável e está diretamente relacionada com a concentração focal do projetor e tamanho da tela. Nos testes realizados a tela esta disposta à uma distância entre 0,80m e 1,20m do projetor, e a holografia podendo ser observada de 0,40m a 2m de distância da tela. As fotos da Figura 5.1 foram tiradas a uma distância entre 0,50m e

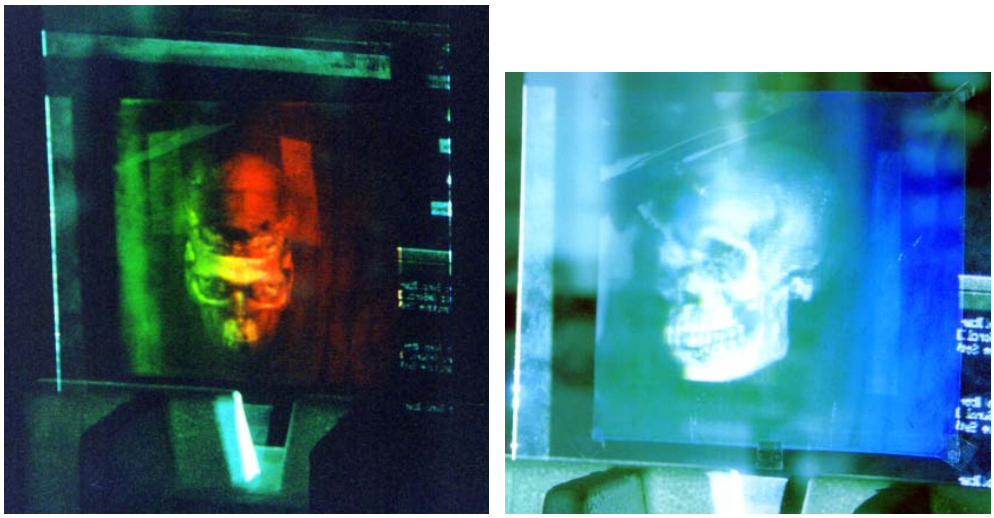


Figura 5.1: Resultado da visualização de estereograma holográfico no Holoprojetor 3.0.

0,80m da tela. O Holoprojetor em funcionamento pode ser visto na Figura 5.2.

Nota-se que a imagem 3D gerada na tela holográfica apresenta uma qualidade visual bem melhor do que a imagem 2.5D gerada na tela do computador. Além de fornecer mais indicadores de profundidade, ela amplia os detalhes vistos na imagem 2.5D. Infelizmente este resultado não pode ser ilustrado em papel.

A projeção sobre a tela holográfica também pode ser observada com o auxílio de óculos vermelho-azul. Neste caso poderão ser observadas duas vistas apenas, com profundidade maior e com paralaxe horizontal discreta, tornando bastante perceptível o salto entre elas.

Com relação às versões anteriores do Holoprojetor, podemos dizer que houve uma melhora substancial de qualidade de imagem. As Figuras 3.4 e 3.7 são respectivamente os resultados obtidos com as versões 1.0 e 2.0 do Holoprojetor. Note que a resolução da imagem na versão 3.0 está limitada apenas à resolução do projetor LCD, que na maioria dos casos é de 800×600 pixels. O tamanho da tela holográfica também pode representar um incômodo para o observador. Durante a maioria dos experimentos foi usada uma tela com dimensão $15 \times 15,5 cm^2$, o que dificultou a concentração do foco do projetor.

Um outro aspecto avaliado neste trabalho foi a melhora de qualidade de imagem devido à interpolação baseada na forma usando a métrica Euclideana em vez da métrica de Chamfer. Esta comparação foi feita usando como referência o melhor caso para a interpolação linear na cena original.

Em situações onde o brilho dos voxels não reflete a anatomia dos tecidos, como é o caso de estruturas ósseas em imagens de ressonância magnética (MRI), a técnica de interpolação baseada na forma é a única alternativa. Já no caso de estruturas ósseas em imagens de tomografia de Raios-X (CT), o brilho dos voxels reflete a anatomia do tecido,

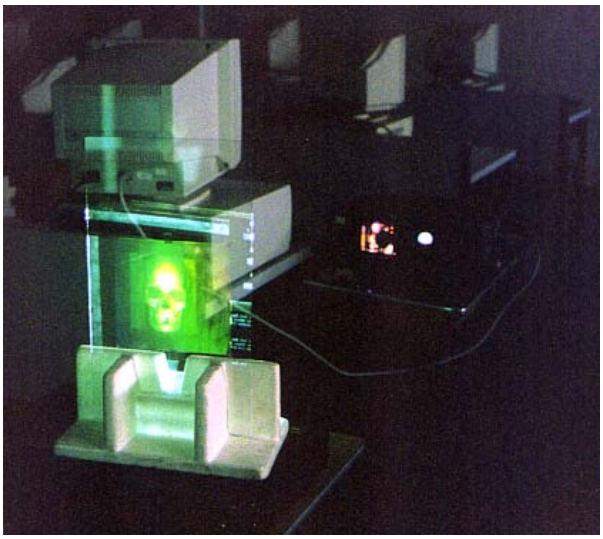


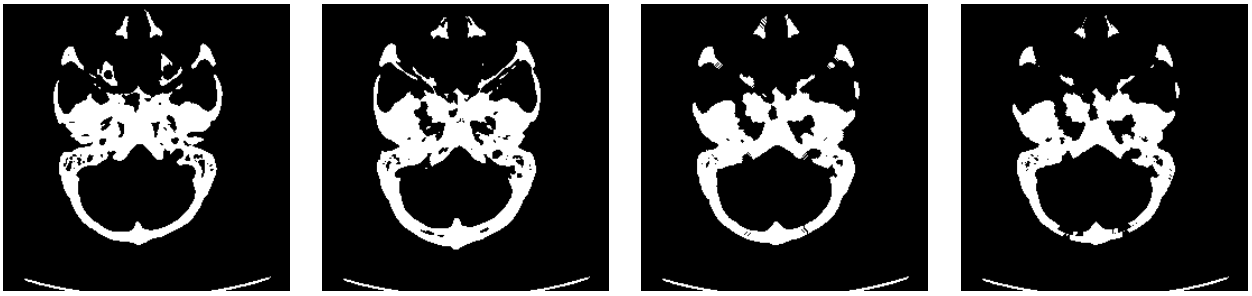
Figura 5.2: Imagem do Holoprojetor 3.0 em funcionamento.

e portanto, espera-se que a interpolação linear na cena original forneca uma qualidade de imagem superior a da interpolação baseada na forma.

O primeiro experimento realizado foi a análise 2D da interpolação de uma imagem original retirada do meio de uma seqüência de três imagens consecutivas. Este experimento reflete uma situação atípica, onde o resultado esperado deve ser inferior ao de uma situação real, pois as fatias originais são espaçadas de 3mm, e após a retirada da fatia intermediária, temos duas imagens espaçadas de 6mm. A Figura 5.3a mostra a fatia original que desejamos estimar após limiarização. As Figuras 5.3b-d mostram os resultados da interpolação linear na cena original seguida de limiarização, limiarização seguida de interpolação baseada na forma com métrica Euclideana e limiarização seguida de interpolação baseada na forma com métrica de Chamfer, respectivamente. Em todos os casos o valor de limiarização é fixo. Nota-se que a Figura 5.3b apresenta o melhor resultado entre as três técnicas de interpolação, como esperado, e que a Figura 5.3c apresenta mais detalhes da fatia original que a Figura 5.3d.

Já nos casos 2.5D e 3D, obtivemos um resultado similar entre a interpolação linear com métrica Euclideana e a interpolação linear na cena original. A Figura 5.4a mostra, por exemplo, uma imagem gerada pelo programa 3Dviewnix [1] que utiliza a métrica de Chamfer na interpolação baseada na forma e a técnica de shell rendering para visualização 2.5D. Sob o mesmo ponto de vista, as imagens 2.5D geradas pelo HoloView, usando interpolação baseada na forma com métrica Euclideana e interpolação linear na cena original,

[1] O 3Dviewnix é um *software* de computação de imagens médicas desenvolvido pelo *Medical Image Processing Group, Dept. of Radiology, University of Pennsylvania.*



(a) (b) (c) (d)

Figura 5.3: (a) Imagem 2D original não interpolada, (b) imagem 2D gerada por interpolação linear na cena original seguida de limiarização, (c) imagem 2D gerada por interpolação baseada na forma com métrica Euclideana e (d) imagem 2D gerada por interpolação baseada na forma com métrica de Chamfer.

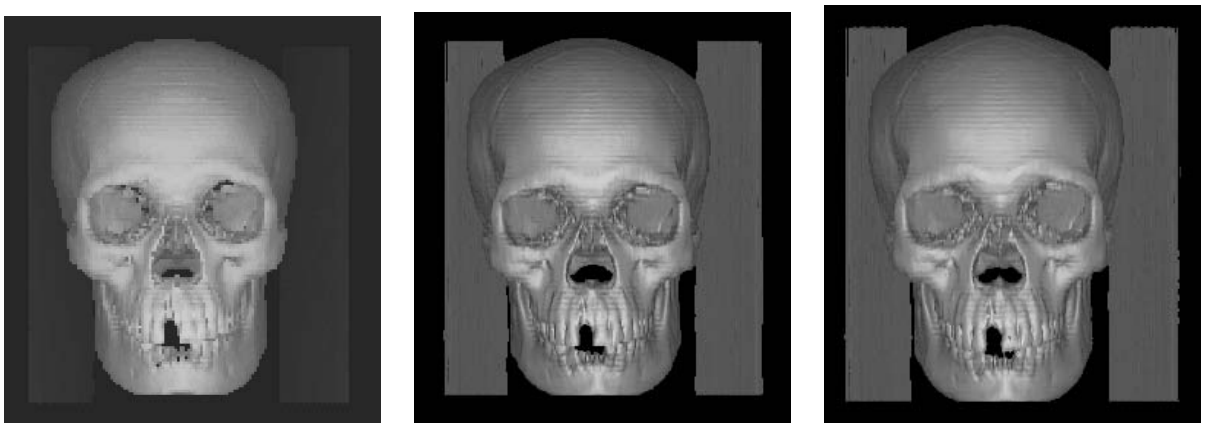
são mostradas nas Figuras 5.4b e 5.4c, respectivamente. Em todos os casos, o vetor normal à superfície, que possui grande influência na qualidade da imagem, foi calculado na cena original após interpolação linear. Observa-se portanto que a interpolação baseada na forma com métrica Euclideana (Figura 5.4b) apresenta uma qualidade de imagem similar ao melhor caso da interpolação linear na cena original (Figura 5.4c). E que ambas apresentam qualidade superior a da interpolação baseada na forma com métrica de Chamfer (Figura 5.4a). Estes resultados são estendidos ao caso de rendering 3D de estereogramas holográficos.

Para finalizar, apresentamos alguns resultados visuais e de manipulação do shell $SShC_4$ para os quatro tipos de rendering apresentados na Tabela 5.4, três exemplos de manipulação do shell $BShJ_1$ para valores diferentes de limiar na mesma cena e tipos de rendering diferentes e também 3 exemplos do shell $SShF_1$, para diferentes valores de tilt e spin.

A Figura 5.5a apresenta o resultado do rendering Normal view de $SShC_4$, na Figura 5.5b é apresentada a imagem de saída do rendering Stereo view e o 3D view é mostrado na Figura 5.5c. A última Figura 5.5d é o resultado da projeção do shell com M.I.P.

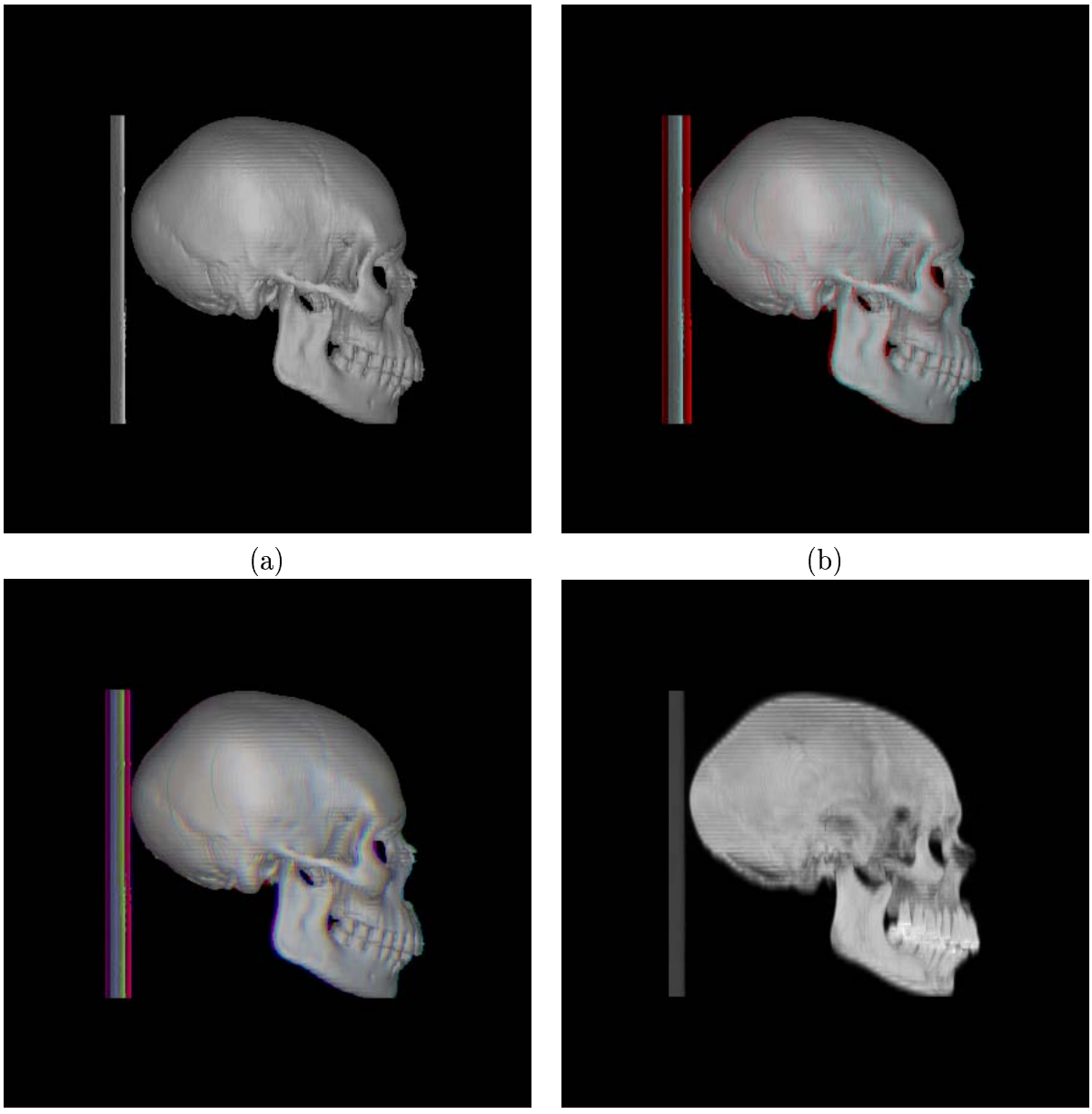
As Figuras 5.6a e 5.6b apresentam o resultado do rendering Normal view de $BShJ_1$ para um limiar de 64 e 103 respectivamente, demonstrando a separação pele e osso *on-the-fly*. A Figura 5.6c apresenta o resultado da projeção deste mesmo shell com M.I.P. Os resultados dos renderings 2.5D de $SShJ_1$ podem ser vistos na Figura 5.7

A Figura 5.8 mostra cortes realizados no shell $SShC_4$ (Figuras 5.8a e 5.8b) e os mesmos cortes realizados no shell BSh_C4 (Figuras 5.8c e 5.8d). Os cortes foram realizados com tilt 180^0, spin 0^0 e corte a 30% e tilt 90^0, spin 90^0 e corte a 40% respectivamente.



(a) (b) (c)

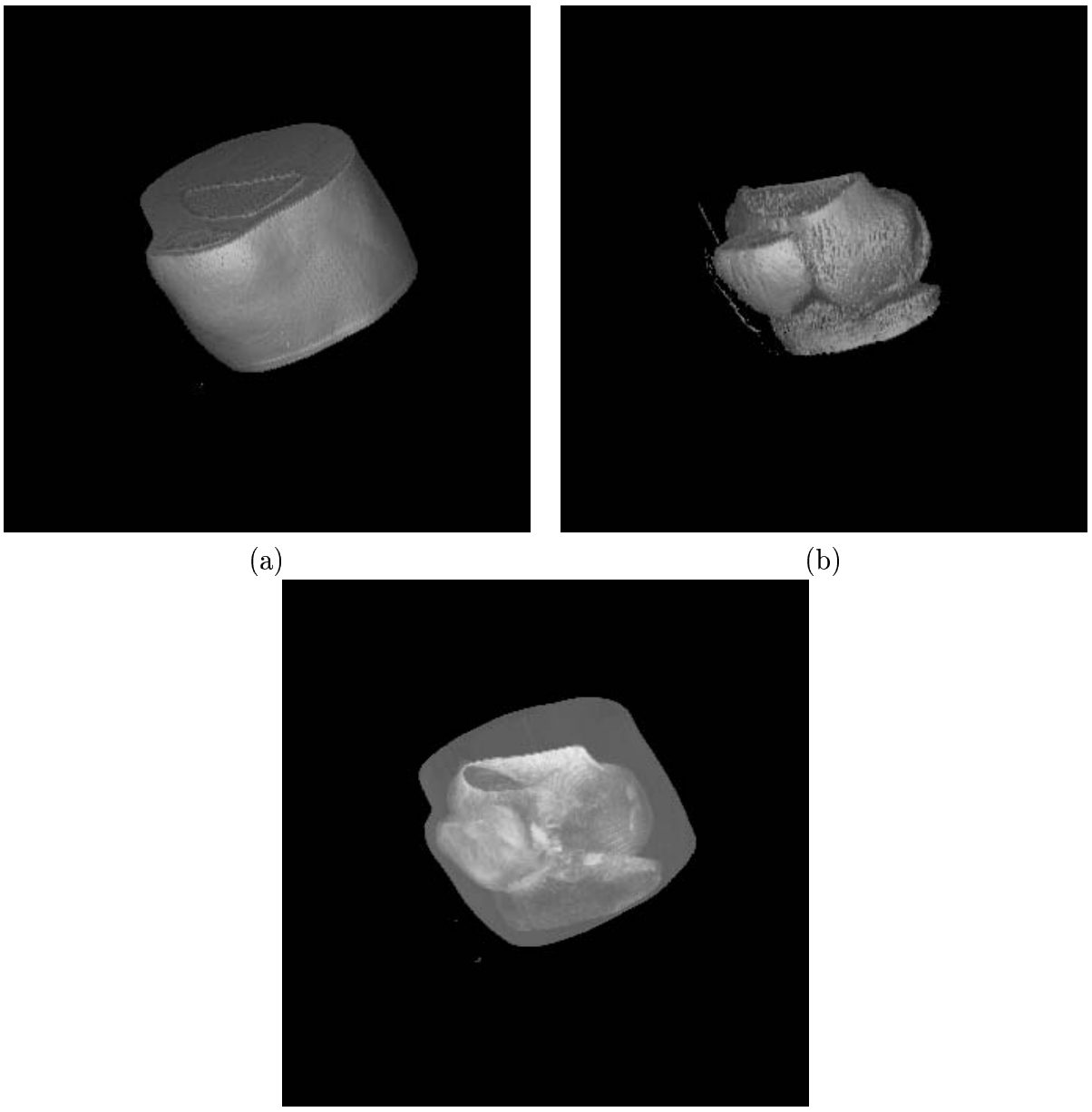
Figura 5.4: (a) Imagem 2.5D gerada no 3Dviewnix utilizando interpolação baseada na forma com métrica de Chamfer, (b) imagem 2.5D gerada no HoloView usando interpolação baseada na forma com métrica Euclideana e (c) imagem 2.5D gerada no HoloView usando interpolação linear na cena original.



(a) (b)

(c) (d)

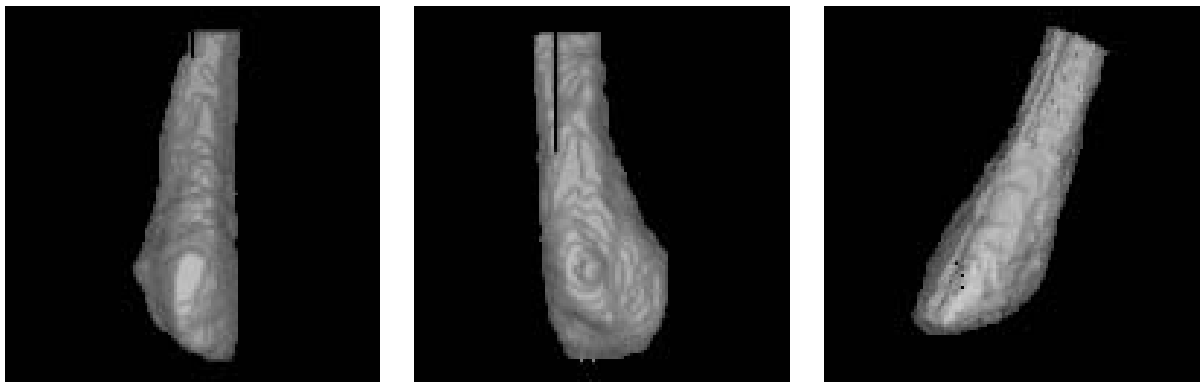
Figura 5.5: Resultado dos renderings de $SShC_4$: (a) Imagem gerada com rendering 2.5D, (b) Imagem de rendering 3D com duas vistas RB, (c) Imagem de rendering 3D com três vistas RGB e (d) Resultado de rendering com técnica de Maximum Intensity Projection.



(a) (b)

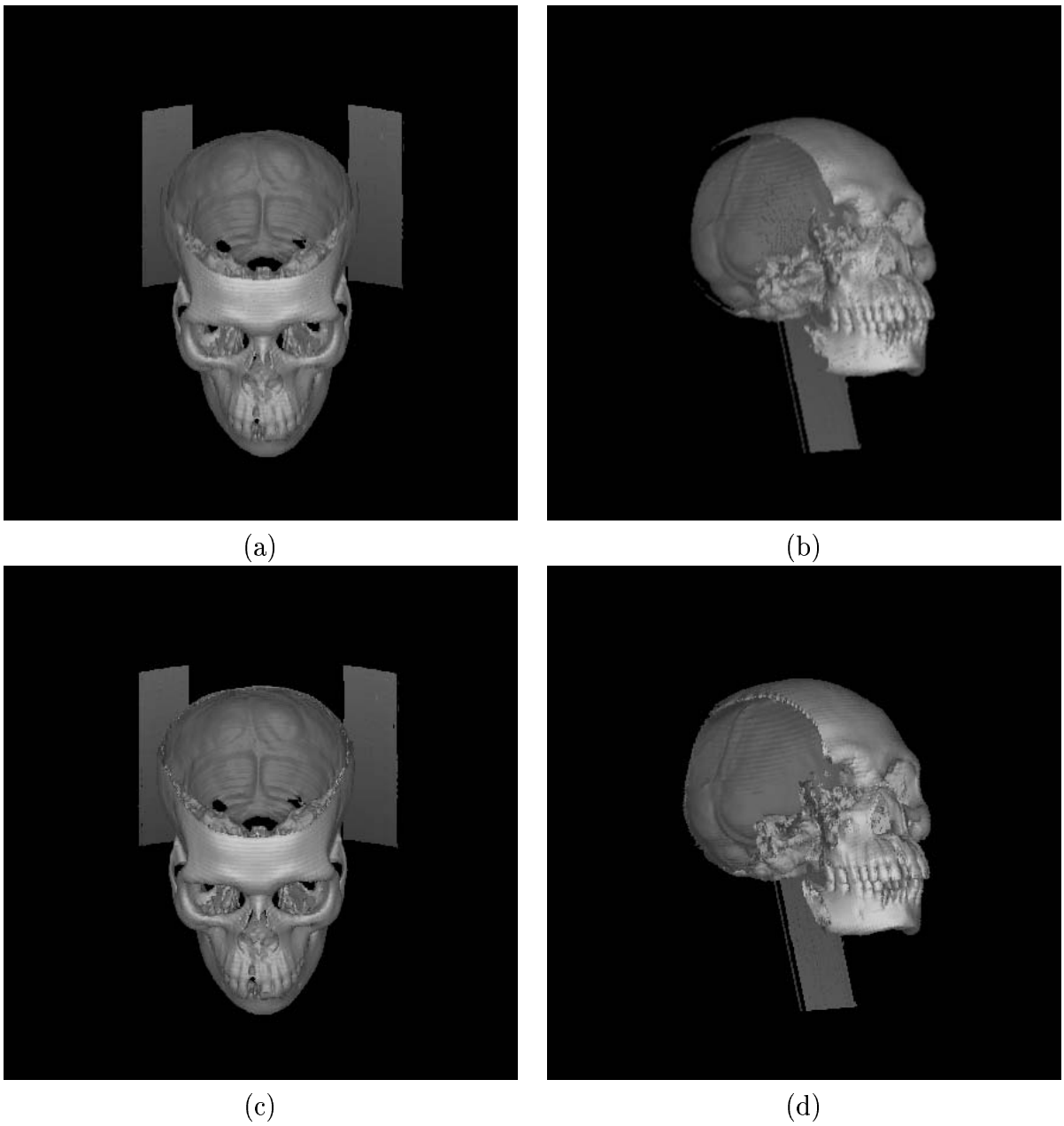
(c)

Figura 5.6: Resultado dos renderings de $BShJ_1$: (a) Imagem gerada com rendering 2.5D e limiar 64, (b) mesmo rendering com limiar 103 e (c) Resultado do rendering com técnica de Maximum Intensity Projection.



(a) (b) (c)

Figura 5.7: Resultado dos renderings de $SShF_1$.



(a) (b)

(c) (d)

Figura 5.8: Resultado de cortes realizados em shell de superfície e shell volumétrico. (a) e (b) são cortes em um shell para rendering de superfícies, (c) e (d) representam o mesmo corte num shell para rendering volumétrico.

Capítulo 6

Conclusão

Neste trabalho mostramos que o Holoprojetor 3.0 é viável para visualização 3D de imagens médicas. Esta viabilidade requer um método rápido de rendering. O método proposto foi o *shell rendering*, que foi modificado para codificar estereogramas holográficos. O sistema final, foi denominado HoloView, o qual implementa várias técnicas de visualização e manipulação de imagens. Foi mostrado também que o uso do Holoprojetor não é limitante, visto que os estereogramas holográficos podem ser visualizados com óculos vermelho-azul. O trabalho foi aprensentado no SPIE Medical Imaging em fevereiro de 1999, San Diego, CA, recebendo a premiação de Menção Honrosa.

Podemos concluir que o rendering de superfícies usando shell rendering apresenta tempos de processamento pequenos o suficiente para visualização *em tempo real*, mesmo quando trabalhamos com renderings triplos RGB para a composição dos estreogramas holográficos. A utilização da interpolação baseada na forma com métrica Euclideana, também representou uma contribuição do ponto de vista da qualidade de imagem final. A qualidade da imagem projetada no Holoprojetor 3.0 é suficiente para torná-lo um sistema alternativo de visualização 3D, sendo este limitado apenas pela qualidade do projetor LCD utilizado.

O HoloView possui várias vantagens para a manipulação e análise de imagens médicas. Em medicina, muitas aplicações requerem a obtenção de medidas entre pontos marcados na imagem 2.5D. O que acontece com freqüência é que, depois de marcados os pontos em imagens 2.5D, rotacionando-se a cena, descobre-se que os pontos foram colocados em lugares errados devido a falsa impressão 3D. Estes erros são reduzidos com a visualização 3D real através do uso do Holoprojetor ou mesmo dos óculos. Outra vantagem é a sensação volumétrica conseguida na visualização das imagens médicas, que facilita o entendimento das estruturas de interesse.

Uma extensão deste trabalho seria a análise da combinação de duas estruturas com opacidades diferentes, como por exemplo a pele e osso de um paciente, para a geração

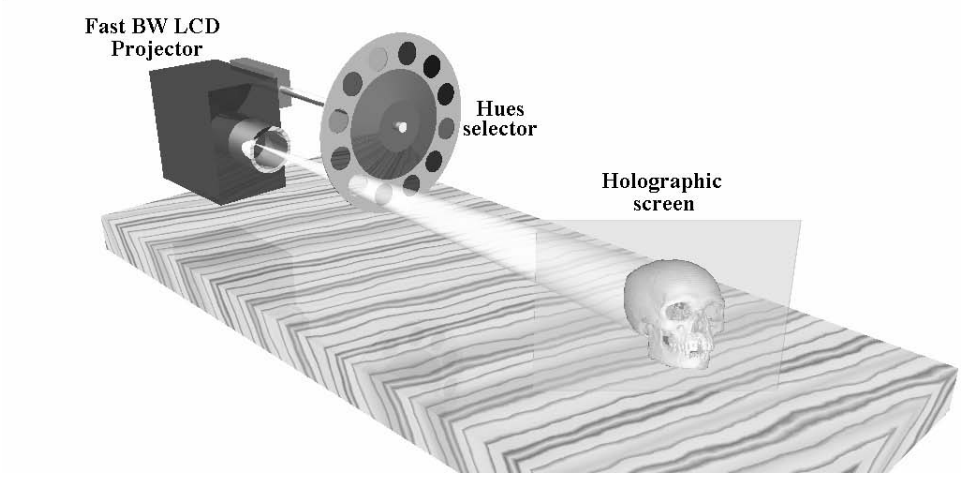


Figura 6.1: Holoprojetor 4.0

de estereogramas holográficos. À pele poderia ser associada uma opacidade de 30% e aos ossos uma opacidade de 70%, através da geração de dois shells e sua posterior combinação. Atualmente não sabemos se uma projeção semi-transparente sobre a tela holográfica apresenta os mesmos bons resultados que uma projeção completamente opaca, devido a diferença das vistas que seria mais acentuada no primeiro caso. Outro trabalho futuro seria a comparação da exatidão das medidas tomadas em sistemas de visualização 2.5D com relação à visualização 3D.

Como extensão do Holoprojetor 3.0, temos a versão 4.0 do Holoprojetor que nunca foi implementada (ver Figura 6.1). Este Holoprojetor codifica mais que três vistas utilizando outras cores puras e um projetor LCD monocromático de alta velocidade. Se observarmos esta projeção de imagens com vistas codificadas em cores sem o auxílio da tela holográfica, teremos a mesma visão de quando assistimos a um filme de um cinema 3D e não utilizamos os óculos vermelho-azul.

O desenvolvimento deste prótipo de Holoprojetor permite a codificação de mais vistas num mesmo rendering, gerando uma paralaxe horizontal praticamente contínua e uma visualização mais nítida das estruturas 3D. Melhorando ainda mais a solução de problemas, como cálculo de distâncias entre pontos, conforme descrito acima. Outra possibilidade seria utilizar óculos com shuter para obtenção de imagens coloridas e com profundidade, o que não é possível mesmo na versão 4.0 do Holoprojetor.

Outras ferramentas que utilizam a interface 3D do Holoprojetor 3.0 também podem ser desenvolvidas. Atualmente o ponteiro do mouse e a própria interface do HoloView são 2D. A passagem do mouse sobre a projeção do estereograma holográfico causa um certo desconforto visual, confundindo as referências de profundidade. Quanto ao resto da interface, ser projetada sobre a tela holográfica em 2D não causa nenhuma desvantagem

e também nenhuma vantagem. O desenvolvimento de uma interface completa para este sistema, desde o ponteiro do mouse até a interface de gerenciamento de janelas pode contribuir para a total aceitação do monitor holográfico em substituição ao convencional. Aplicações nas áreas de engenharia, matemática, química, medicina e computação gráfica podem receber amplos benefícios com a utilização deste sistema.

Bibliografia

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1993.

[2] E. Bertini. Um Sistema para Visualização Holográfica. Master's thesis, Instituto de Computação, UNICAMP, Março 1998.

[3] E. Bertini, C. F. X. de Mendonça N., J. J. Lunazzi, and P. L. de Geus. Um Sistema para Visualização Holográfica. *Anais do IX SIBGRAPI'96*, pages 23–29, October 1996.

[4] G. P. Carnielli. Um Método Rápido para Rendering de Volumes com Perspectiva Digital. Master's thesis, Instituto de Computação, UNICAMP, Dez 1999.

[5] A.R. Cohen, B. Roysam, and J.N. Turner. Automated tracing and volume measurements of neurons from 3D confocal fluorescence microscopy data. *Journal of Microscopy*, 173(2):103–114, Feb 1994.

[6] O. Cuisenaire and B. Macq. Fast Euclidean distance transformation by propagation using multiple neighborhoods. *Computer Vision and Image Understanding*, 76(2):163–172, Nov 1999.

[7] E. G. da Fonseca. Dois Sistemas para Animação Holográfica. Master's thesis, Instituto de Computação, UNICAMP, Março 1998.

[8] E. G. da Fonseca, P. L. de Geus, C. F. X. de Mendonça N., J. J. Lunazzi, and E. Bertini. A Holographic Visualization System: A Sequel. *in Proc. of The XI International Symposium on Computer Graphics, Image Processing and Vision, SIBGRAPI'98*, pages 135–141, October 1998.

[9] C. F. X. de Mendonça N., A. X. Falcão, C. A. Vannini, and J. J. Lunazzi. Fast Holographic Stereograms Display Using Shell Rendering and a Holographic Screen. *in Proceedings of SPIE on Medical Imaging*, pages 484–492, February 1999.

[10] M. Diamand. Sistema para Visualização Holográfica de Figuras Geradas por Computador. Master's thesis, Faculdade de Engenharia Elétrica, UNICAMP, Dezembro 1994.
http://www.geocities.com/CapeCanaveral/Lab/6146/tese_md.html.

[11] A. X. Falcão. Visualização de Volumes Aplicada à Área Médica. Master's thesis, FEEC, UNICAMP, Fevereiro 1993.

[12] A.X. Falcão, R.A. Lotufo, and G. Araujo. The image foresting transformation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1999. submetido em dezembro de 1999.

[13] A.X. Falcão, J.K. Udupa, S. Samarasekera, S. Sharma, B.E. Hirsch, and R.A. Lotufo. User-steered image segmentation paradigms: live-wire and live-lane. *Graphical Models and Image Processing*, 60(4):233–260, Jul 1998.

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

[15] G. Frieder, D. Gordon, and R. A. Reynolds. Back-to-front display of voxels based objects. *IEEE Computer Graphics and Applications*, 5(1):52–60, Jan 1985.

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

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

[18] A. Goshtasby and D.A. Turner. Segmentation of cardiac cine MR images for extraction of right and left ventricular chambers. *IEEE Transactions on Medical Imaging*, 14(1):56–64, Mar 1995.

[19] G.J. Grevera, J.K. Udupa, and D. Odhner. One order of magnitude faster isosurface rendering in software on a PC than using dedicated, general purpose rendering hardware. *in Proceedings of SPIE on Medical Imaging'99*, 3658:202–211, Feb 1999.

[20] W.E. Higgins and E.J. Ojard. Interactive morphological watershed analysis for 3D medical images. *Computerized Medical Imaging and Graphics*, 17(4/5):387–395, 1993.

[21] A. Kalvin. *Segmentation and surface-based modeling of objects in three-dimensional biomedical images.* PhD thesis, Dept. of Computer Science, New York University, Mar 1991.

[22] P. Lacroute and M. Levoy. Fast volume rendering using a shear-warp factorization of viewing transformation. *Computer Graphics*, 28(4):451–458, 1994.

[23] J. J. Lunazzi. New possibilities in the utilization of holographic screen. *Proc. of the SPIE meeting Electronic Imaging, conference Practical Holography VI, San Jose-CA-USA*, 9-14:289–293, February 1992.
http://www.geocities.com/CapeCanaveral/Lab/6416/spie92.pdf.

[24] J. J. Lunazzi. patent Procédé et dispositif pour projeter et observer des images differenciees ou stereoscopiques, dessins, photographies, films cinematographiques ou video. **INPI-FR**, N 8907241, 1992.
http://www.geocities.com/CapeCanaveral/Lab/6146/patentfr.html.

[25] S.P. Raya and J.K. Udupa. Shape-based interpolation of multidimensional objects. *IEEE Transactions on Medical Imaging*, 9:32–42, 1990.

[26] R.C. Rhoad, J.J. Klimkiewicz, G.R. Williams, S.B. Kesmodel, J.K. Udupa, B. Kneeland, and J.P. Iannotti. A new in vivo technique for 3D shoulder kinematics analysis. *Skeletal Radiology*, 27:92–97, 1998.

[27] K.R. Subramanian and D.S. Fussell. Applying space subdivision techniques to volume rendering. *Proceedings of Visualization'90*, pages 150–159, 1990.

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

[29] J.K. Udupa, B.E. Hirsch, S. Samarasekera, H. Hillstrom, G. Bauer, and B. Kneeland. Analysis of in vivo 3D internal kinematics of the joints of the foot. *IEEE Transactions on Biomedical Engineering*, 45:1387–1396, 1998.

[30] M.W. Vannier, J.L. Marsh, and J.O. Warren. Three-dimensional computer graphics for craniofacial surgical planning and evaluation. *Computer Graphics*, 17:263–274, Jul 1983.