**Dentes Virtuais: Visualização
para Apoio a Atividades Odontológicas**

Milton Hirokazu Shimabukuro

Orientação: Profa. Dra. Rosane Minghim

*Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação –
USP, como parte dos requisitos para obtenção do título de Mestre em Ciências –
Área de Ciências de Computação e Matemática Computacional*

*São Carlos- SP
Junho de 1998*

998806

Dedico este trabalho
Aos meus filhos Victor Eiji e Nicole Meyi
À minha esposa Cristina Satiko
Às minhas irmãs Edina Harumi e Adriana Ruriko
À memória do meu irmão Gilson Hideki
Aos meus pais Hiromoti e Mineko
Aos meus avós Hosei e Matsu

Agradecimentos

À professora Rosane Minghim, orientadora, pelo incentivo, apoio, compreensão e direção na execução deste trabalho.

Às professoras Maria Cristina e Agma, do Grupo de Computação Gráfica e Processamento de Imagens do SCE-ICMC-USP, pela gentileza e prontidão com que sempre me atenderam.

Aos colegas do Grupo de Computação Gráfica e Processamento de Imagens, Sandra, Veridiana, Pedro, André, Rogério, Paulo Renato, Eldes e Magnus, pelo companheirismo.

Aos amigos Silvio, Antonio Marcos Shiro, Rogério e Ronaldo, com os quais compartilhei as alegrias e superei as adversidades desta caminhada.

Aos colaboradores da SYS-UEA-Inglaterra, Andy Day e Mike Goetz.

À professora Lizeti Ramalho, da Unesp de Araraquara, pelo fornecimento de dados.

Ao amigo Celso Kenji Nishiyama, do Centrinho-Bauru, pelas informações.

Ao professor Messias Meneguette Jr., da FCT-Unesp de Presidente Prudente, que muito colaborou para o início e a concretização desta jornada.

Ao professor Antonio Maria Garcia Tommaselli, da FCT-Unesp, pelo incentivo e pela disposição em colaborar sempre que solicitado.

Aos professores Neri Alves, Nilton Imai e Ailton Pagamisse, da FCT-Unesp, pelo apoio e valorização.

Aos professores Klaus e Elisa Schlünzen, da FCT-Unesp, pelo incentivo e cooperação.

Aos professores Maurício Galo, Maria de Lourdes e Mônica, da FCT-Unesp, pelo apoio e entusiasmo.

Aos professores Octavio Freire e João Fernando, da FCT-Unesp, pela colaboração.

Aos amigos José Angelo, Paulo Roberto, Patrícia Scalco e Ricardo Barbosa pelo coleguismo.

À minha esposa Cristina, pela cumplicidade e pela colaboração neste trabalho.

Aos funcionários do ICMC-USP e da FCT-Unesp, que sempre se mostraram dispostos na prestação de seus serviços.

À USP e à Unesp, pela oportunidade.

ÍNDICE

Capítulo 1. INTRODUÇÃO	01
Capítulo 2. PRÉ-PROCESSAMENTO	05
2.1 Noções de Operações de Pré-Processamento	07
2.2 Registro (Registration) e Restauração (Restoration)	08
2.2.1 Transformação Isogonal	09
Capítulo 3. RECONSTRUÇÃO DE MODELOS 3D	12
3.1 Aspectos da Triangulação entre Contornos	13
3.2 Métodos de Reconstrução de Superfícies	15
3.3 Construção de Superfícies Utilizando Funções de Campo	17
3.3.1 Aplicação da Função de Campo	19
Capítulo 4. VISUALIZAÇÃO PARA MODELOS 3D	23
4.1 Visualização Volumétrica	25
4.1.1 Uma Terminologia para Visualização Volumétrica	26
4.1.2 Voxels e Células	27
4.1.3 Formas de Criação da Imagem	28
4.1.4 Contour-connecting	28
4.1.5 Opaque Cube ou Cuberille	29
4.1.6 Marching Cubes, Marching Tetrahedra e Dividing Cubes	29
4.1.7 Ray-casting	31
4.2 Interação	32
4.2.1 Utilização de Metáforas	32
4.2.2 Representação Tridimensional	33
4.2.3 Realidade Virtual (VR - Virtual Reality)	35
Capítulo 5. RECONSTRUÇÃO DO DENTE VIRTUAL - UMA IMPLEMENTAÇÃO	37
5.1 Características do VTK	38
5.2 Estrutura do Projeto do Dente Virtual	39
5.3 Elaboração do Algoritmo	42
5.3.1 Aspectos e Elementos da Implementação do Filtro	44
5.3.1.1 Determinação do Posicionamento Relativo	45

5.3.1.2 Cálculo da Menor Distância	47
5.3.1.3 Fechamento de Contornos.....	49
5.4 Lógica e Detalhes do Filtro Implementado	50
5.5 Funcionalidade de Programas Complementares.....	52
5.5.1 Extração Manual de Contornos e Referenciais	53
5.5.2 Correção Geométrica	54
5.5.3 Geração de Arquivo no Formato VTK	55
5.5.4 Interface para Realizar a Reconstrução.....	55
Capítulo 6. RESULTADOS E CONCLUSÕES	57
6.1 Testes Realizados	57
6.1.1 Reconstrução de Detalhes do Dente.....	59
6.1.2 Análise Interna e Aplicação de Correção Geométrica	61
6.1.3 Comparação com o vtkImplicitModeller	65
6.1.4 Imagens no Formato do VRML.....	68
6.2 Considerações Gerais	69
6.3 Conclusões	71
REFERÊNCIAS BIBLIOGRÁFICAS.....	74
APÊNDICE A - Página do Manual VTK para o Filtro vtkSamplingFromContoursFilter.....	A.1
APÊNDICE B - Formato dos Dados de Entrada para vtkSamplingFromContoursFilter.....	B.1

LISTA DE FIGURAS

1.1	Passos do processo de reconstrução e visualização de um dente	04
2.1	Conjunto de fatias formando um volume de dados.....	06
2.2	Relação entre cena, voxel ou célula, imagem e pixel	06
2.3	Pontos de ligação correspondentes em dois segmentos de imagem.....	09
2.4	Esquema da interpolação de tons de cinza.....	09
3.1	Contorno de uma estrutura em uma fatia	13
3.2	Problemas de tecelagem e ramificação	15
3.3	As fases do algoritmo apresentado em [Jon 94]	18
3.4	Grades bidimensionais formando um volume.....	18
3.5	Resultados do uso de diferentes resoluções e funções de campo.....	19
3.6	Superfícies geradas por diferentes funções de campo.....	20
3.7	Particionamento de uma fatia em setores.....	20
3.8	Exemplo de interpolação no Marching Cubes.....	21
4.1	Terminologia para Visualização Volumétrica	26
4.2	Exemplo de voxel (esquerda) e célula (direita).....	28
4.3	Busca e conexão de contornos.....	29
4.4	Representação de superfície por cubos.....	29
4.5	Numeração dos vértices e formação do índice no Marching Cubes	30
4.6	15 casos do Marching Cubes.....	30
4.7	Exemplo de divisão de uma célula em tetraedros.....	31
4.8	Divisão de uma célula em subcélulas.....	31
4.9	Raio disparado do plano da imagem em direção ao volume	32
4.10	Projeção perspectiva e paralela.....	34
5.1	Contornos em fatias paralelas e igualmente espaçadas	40
5.2	Processo geral de criação de um modelo 3D de dentes	41
5.3	Elementos do teste com vtkImplicitModeller.....	43
5.4	Estrutura de um elemento da lista de arestas e da lista de arestas ativas.....	45
5.5	Relacionamento entre lista de arestas e lista de arestas ativas	46
5.6	Relacionamento entre pontos e contornos.....	47
5.7	Esquema do cálculo de distância na segunda parte do filtro.....	47
5.8	Vértices de um cubo formado por duas fatias adjacentes	48
5.9	Definição da posição da superfície de fechamento	49
5.10	Lógica da fase de determinação da posição relativa dos pontos da grade	50
5.11	Lógica da fase de cálculo de distâncias por onde a superfície passa	50
5.12	Lógica para o fechamento da superfície	50
5.13	Exemplo de uso do filtro vtkSamplingFromContoursFilter em C++	52
5.14	Interface gráfica do programa de extração manual de contornos e referenciais.....	53
5.15	Interface gráfica do programa de correção geométrica.....	54

5.16	Interface gráfica do programa de geração de arquivo no formato VTK	55
5.17	Interface gráfica do programa de interface para reconstrução	56
6.1	Desenho dos contornos, à mão, digitalizado.....	59
6.2	Reconstrução com os dados do conjunto 1.....	60
6.3	Reconstrução com e sem fechamento da superfície.....	60
6.4	Dente em molde com hastes de grafite e contornos correspondentes.....	61
6.5	Resultados da reconstrução usando dados do conjunto 2.....	63
6.6	Reconstrução empregando vtkSamplingFromContoursFilter e vtkImplicitModeller.....	67
6.7	Detalhes de imagens vistas no ambiente do VRML e do VTK	68
6.8	Configuração da junção de duas arestas num polígono	70
6.9	Modelo 3D de dente reconstruído usando o vtkSamplingFromContoursFilter	71
B.1	Exemplo de um dataset válido para o vtkSamplingFromContoursFilter.....	B.2

LISTA DE TABELAS

6.1	Descrição dos conjuntos de dados utilizados nos testes	58
6.2	Resultados obtidos pela variação da resolução de amostragem	62
6.3	Relação entre os valores da tabela 6.2.....	62
6.4	Parâmetros da transformação isogonal relativos ao conjunto de dados 2	64
6.5	Dados da reconstrução sem e com correção geométrica	65
6.6	Resultados dos testes envolvendo <code>vtkImplicitModeller</code> e <code>vtkSamplingFromContoursFilter</code>	65
6.7	Comparação entre valores presentes na tabela 6.6	66

Resumo

Neste projeto de mestrado foi estudado um processo de geração de modelos tridimensionais de dentes a partir de dados reais. Foram elaborados e implementados um algoritmo para reconstrução 3D de dentes, e vários programas complementares para apoiar a reconstrução. Um modelo tridimensional é criado a partir de contornos planares que descrevem as estruturas de interesse, representados por polígonos extraídos de fatias ou cortes paralelos de um dente. O algoritmo foi implementado na forma de uma classe, em extensão a uma biblioteca de visualização de propósito geral, o VTK (*Visualization Toolkit*), com resultados bastante promissores.

Abstract

In this MSc research a process of generating three-dimensional models for teeth from real data was studied. An algorithm for 3D tooth reconstruction and several complementary programs to aid reconstruction were designed and implemented. A three-dimensional model is created from planar contours describing the structures of interest, represented by polygons extracted from parallel slices of a tooth. The algorithm was implemented as an object class, extending a general-purpose visualization library called VTK (Visualization Toolkit), with very promising results.

1. INTRODUÇÃO

A Visualização Científica usa técnicas da Computação Gráfica para ajudar cientistas e outros profissionais a capturar o significado dos seus dados, facilitando o entendimento de entidades físicas e a interpretação de resultados de experimentos e medições. A Visualização tem se desenvolvido muito nos últimos anos, a partir de uma iniciativa norte-americana, que reconheceu essa disciplina como uma área separada, devido às suas características próprias em relação à Computação Gráfica convencional e também às disciplinas que visa apoiar [CG 87]. A partir desse reconhecimento, grande desenvolvimento e ampliação de atuação da Visualização têm sido observados. A aplicação em imagens médicas está entre as principais áreas beneficiadas.

Os sistemas de visualização permitem a extração de informações contidas em descrições numéricas de fenômenos complexos através do uso de sistemas gráficos interativos. Tais sistemas são usados tanto como ferramenta de análise de informação, quanto como ferramenta para comunicação de resultados [Elv 94]. Aplicações na área científica incluem imagens médicas, simulações em dinâmica dos fluidos, meteorologia, biologia molecular, dados ambientais, cartografia, astronomia, e inúmeras outras. A tecnologia advinda de pesquisas em Visualização Científica tem sido utilizada em muitas outras áreas, como aplicações financeiras e visualização de dados em geral.

A aplicação de técnicas de visualização 3D em Medicina tem concentrado grande parte do esforço dedicado à área, tanto em sistemas de apoio a diagnósticos, como o apoio ao entendimento da anatomia e de doenças humanas [Udu 91] [Cos 96] [Tre 96] [Lor 95] [Rou 94]. Na área médica, os dados são, via de regra, gerados por dispositivos de medição como Raios-X, Tomografia Computadorizada (CT - *Computerized Tomography*) e Ressonância Magnética (MRI

- *Magnetic Resonance Imaging*). O resultado dessas medições são representados por imagens bidimensionais com resolução e clareza variáveis. Por um processo denominado “reconstrução” [Lor 87] [Mat 94] [Mon 94] [Alg 96], modelos tridimensionais desses dados podem ser gerados, possibilitando uma visão mais global e real dos órgãos e suas partes apresentadas nos exames, e também um melhor entendimento da estrutura interna dos órgãos, além do benefício educacional. O projeto do Ser Humano Visível (*Visible Human Project* - [Tre 96] [Lor 95]) se destaca como uma iniciativa na criação de um modelo 3D completo do corpo humano, a partir de imagens obtidas de cortes efetivos de cadáveres.

Existem algumas dificuldades na utilização de modelos e imagens 3D para aplicações médicas. Do ponto de vista tecnológico, existem dificuldades na criação de modelos precisos. Adicionalmente, existe resistência dos profissionais da área na utilização desses novos métodos visuais, que variam desde o uso de cores até a confiabilidade do modelo “artificialmente” elaborado, bem como problemas na própria redução de dimensão necessária para a apresentação em dispositivos 2D, e a questão da geração de imagens complexas em tempo real.

Em Visualização existem basicamente dois métodos de geração de imagens 3D a partir de dados de cortes (fatias) 2D justapostos em um volume. O primeiro deles é utilizar uma representação intermediária, uma superfície, resultante do processo de reconstrução mencionado acima. Essas superfícies podem ser representadas por poligonais e passar pelo *rendering* por métodos convencionais da Computação Gráfica. O outro método, atualmente denominado *rendering* volumétrico direto (DVR - *Direct Volume Rendering*) [Kau 93], calcula parâmetros de iluminação diretamente de unidades de volume (voxels - os correspondentes 3D dos pixels em 2D). Tais parâmetros são aproximados, e as várias superfícies não podem ser manipuladas individualmente. O *rendering* volumétrico direto possui resultados visuais muito efetivos, mas a um custo computacional muito alto. *Rendering* por superfícies (*Surface Rendering*) permite acesso direto a superfícies individuais, mas é sujeita a mais erros de geração, decorrentes de exceções no processo de triangulação envolvido [Eko 91], além de não permitir a descrição do interior dos objetos.

Vários métodos de geração de visualização por superfícies existem, mas o problema continua em aberto, uma vez que os métodos de reconstrução continuam a ser aprimorados para cobrir efetivamente todos os possíveis casos de diferenças entre cortes paralelos das imagens originais, e portanto, resolver os problemas de ambiguidades [Mat 94], ramificações entre cortes vizinhos [Eko 91], e representação de dados irregulares [Alg 96].

A imensa capacidade dos equipamentos de detecção e medição tem motivado maiores avanços nas técnicas visuais de representação. Em odontologia, essa capacidade tem tomado forma e

chamado a atenção em nível nacional [Fer 96]. Possíveis aplicações da representação 3D de dentes e outras partes da boca e face são inúmeras. O primeiro benefício é educacional, ou seja, utilizar um modelo computacional para explicar estruturas e funções dentárias, tanto em nível escolar como de estudo universitário não-intervencionista (educação do dentista anterior à aproximação com o ser humano ou mesmo com cadáveres humanos). Um outro sub-produto educacional é a explicação, a nível de consultórios e clínicas, ao usuário de serviços dentários (pacientes), dos procedimentos clínicos a serem realizados. Fazer isso com o uso de um modelo virtual de dentes e boca deve, em geral, tranquilizar o paciente quanto ao tratamento, sem necessidade de ilustrações agressivas, como as apresentadas em livros, revistas e vídeos de procedimentos reais. Uma outra utilidade de um modelo virtual criado, a partir de dados reais de dentes, é no planejamento de procedimentos e estudo e simulação da evolução de doenças. Uma outra aplicação importante é no treinamento de cirurgias e outros procedimentos delicados.

Várias limitações existem para a implementação dessas idéias no atual estado da arte da visualização de objetos em procedimentos odontológicos. Essas dificuldades variam desde limitações referentes à obtenção de dados (não é tão simples realizar *scans* de vários cortes de um mesmo dente), até a utilização de sistemas de visualização (normalmente caros, difíceis de usar para usuários leigos, e não direcionados a esse propósito) em nível doméstico ou de consultórios.

Existem algumas semelhanças e diferenças no desenvolvimento de visualizações para aplicações médicas e odontológicas. A maioria das visualizações na área médica se refere ao apoio a diagnósticos. Enquanto essa aplicação ainda é limitada em Odontologia, o estudo, a educação e o planejamento de procedimentos são áreas muito promissoras. A reconstrução do modelo dentário, entretanto, segue as mesmas linhas das reconstruções médicas, porém com requisitos diferentes de precisão e eficiência.

Alguns esforços vêm sendo empregados na geração de modelos tridimensionais para aplicações odontológicas. Tais trabalhos incluem reconstruções de mandíbulas [Sei 95], simulação do processo de mastigação [Mys 95], visualização 3D de forças aplicadas sobre dentes e seus efeitos [Sch 96a], e análise tridimensional de formação e erupção de dentes [Kre 96][Bro 96]. A área é ampla e as aplicações inúmeras, com muitos itens necessitando novas pesquisas e novas soluções.

O objetivo do presente projeto de mestrado é elaborar e implementar um algoritmo que permita a reconstrução de modelos 3D de dentes, e construir um esquema de visualização desses modelos possibilitando a exibição e exploração de suas estruturas, internas e externas. O modelo 3D de um dente é reconstruído a partir de um conjunto de contornos planares, presentes em cortes de um dente e descrevendo as estruturas de interesse.

O produto deste mestrado é um dos primeiros passos de um projeto em desenvolvimento no SCE-ICMC¹ em colaboração com a SYS-UEA² e com a FOA-UNESP³, para tratamento e implementação de visualização tridimensional de informações odontológicas. Isso inclui estudo de algoritmos apropriados para reconstrução de estruturas presentes na boca e arcada dentária, técnicas gráficas e sonoras de exploração de tais estruturas, e interfaces multimídia adequadas para análise, treinamento e apresentação de informações dentárias anatômicas, cirúrgicas, e patológicas.

A criação de um modelo 3D a partir de dados reais passa por uma série de estágios, ilustrados na figura 1.1.

Figura 1.1 Passos do processo de reconstrução e visualização de um dente.

Após a aquisição e possível digitalização, os dados são preparados (P2) para a geração de um modelo tridimensional (P3) do objeto mapeado, o qual é exibido e explorado pelo usuário (P4).

A parte prática deste trabalho incluiu a implementação de um algoritmo de reconstrução a partir de contornos em fatias paralelas aplicado ao caso dentário. Além disso, programas de apoio às demais etapas do processo (extração de contornos, correção geométrica, formatação de dados e interface para executar a reconstrução e a visualização) foram desenvolvidos. Utilizando essas ferramentas, modelos de dentes foram reconstruídos, visualizados e exportados para o formato VRML para exploração. Os resultados foram bastante satisfatórios, sendo que a reconstrução recuperou a forma 3D original com um alto grau de fidelidade, demonstrando a aplicabilidade da abordagem selecionada ao caso das aplicações em odontologia.

Os capítulos 2, 3 e 4 desta dissertação abordam, respectivamente, os temas de pré-processamento, reconstrução e visualização (etapas correspondentes aos passos P2, P3 e P4 da figura 1.1). Aspectos da implementação estão detalhados no capítulo 5 e, os resultados e avaliações, bem como as conclusões deste trabalho são apresentadas no capítulo 6.

¹ Departamento de Ciências de Computação e Estatística - Instituto de Ciências Matemáticas e de Computação - Universidade São Paulo - Campus de São Carlos.

² School of Information System - University of East Anglia - Inglaterra.

³ Faculdade de Odontologia de Araraquara - Universidade Estadual Paulista - Campus de Araraquara.

2. PRÉ-PROCESSAMENTO

Em diversas situações, os dados passam por uma etapa de preparação para as fases posteriores de reconstrução e visualização, podendo ser transformados para atender a um formato de entrada, para melhorar a eficiência de processamento ou ainda para minimizar a necessidade de recursos computacionais. Um exemplo é ilustrado por situações em que cada imagem é adquirida por partes, requerendo um processo de junção dessas partes, formando um mosaico, para compor cada fatia [Mon 94a]. Outro exemplo é apresentado em [Doy 93], no qual as imagens sofrem um processamento para separar a estrutura de interesse (contorno externo do tecido embrionário) da área circundante, com as coordenadas do contorno sendo armazenadas separadamente.

Cada imagem que é obtida através de algum processo de aquisição, como Tomografia Computadorizada, representa uma fatia ou uma seção do objeto amostrado. A justaposição ordenada de fatias forma um conjunto tridimensional, que constitui, assim, um volume de dados (figura 2.1). Frequentemente esse volume precisa passar por algumas transformações para estar adequado ao uso nas etapas seguintes. Essas transformações podem ser realizadas em cada fatia individualmente ou no volume como um todo. A extração de contornos de objetos presentes em uma fatia é um exemplo do primeiro caso, enquanto a classificação de diferentes materiais presentes num sub-volume [Lai 94] se encaixa no segundo caso.

O procedimento de preparação pode ser automático ou semi-automático. Especialistas nas áreas de aplicação podem ser solicitados para auxiliar durante o procedimento ou para validar algum resultado.

Figura 2.1 Conjunto de fatias formando um volume de dados.

Para compatibilidade com as referências consultadas, são definidos os seguintes termos:

- a) Imagem: Matriz de valores que definem um objeto gráfico na tela (2D) do computador;
- b) Pixel: Unidade da imagem ou tela de computador;
- c) Cena: Conjunto de objetos 3D que compõe uma configuração a ser visualizada ou renderizada;
- d) Voxel ou Célula: Cada um dos componentes unitários de um volume. Voxels representam valores escalares armazenados no interior da unidade de volume, enquanto células armazenam vários valores escalares, cada um associado a um de seus vértices;
- e) Intensidade: valor associado a um pixel;
- f) Densidade: Valor escalar associado a um voxel, ou ao vértice de uma célula;
- g) "Ponto": Utilizado para referenciar um pixel, um voxel ou um vértice de uma célula.

As figura 2.2 ilustra os termos definidos acima.

Figura 2.2 Relação entre cena, voxel ou célula, imagem e pixel.

Embora voxel e célula sejam dois tratamentos essencialmente diferentes para unidades de volume, muitas operações podem ser realizadas igualmente para ambos. Assim, a distinção entre eles será mencionada apenas quando necessário.

Algumas das formas de aquisição de imagens são: microscópio, Tomografia Computadorizada e Ressonância Magnética. Com os objetivos da aplicação definidos, o conhecimento das características dos diferentes processos de aquisição é importante na escolha da forma de aquisição, na escolha de técnicas de processamento do volume de dados, na interpretação do resultado e na definição da técnica de visualização. De [Udu 91] podemos extrair o seguinte exemplo: em Tomografia Computadorizada, a densidade de um voxel representa a atenuação dos raios pelo tecido que está contido no voxel; em Ressonância Magnética, por outro lado, a densidade representa um agregado de propriedades magnéticas das moléculas do tecido contido no voxel. As densidades são refletidas em intensidades de brilho de pixel quando as imagens são mostradas.

Nas Seções seguintes são apresentadas várias formas de pré-processamento, dentre as quais algumas são abordadas com mais detalhes. As operações aqui apresentadas são independentes entre si, mas o resultado produzido é dependente da sequência de operações adotada, ou seja, transformações sobre o mesmo conjunto de dados aplicadas em sequências diferentes podem levar a resultados diferentes.

2.1 Noções de Operações de Pré-Processamento

Esta Seção apresenta as características gerais de algumas operações que podem ser executadas dentro da fase de pré-processamento:

1. Volume de Interesse (VOI - *Volume of Interest*). O propósito principal de operações de VOI é minimizar o espaço de armazenamento requerido pelo volume de dados. Isso é feito pela criação de um outro volume de dados que contenha toda a estrutura de interesse e o mínimo possível das partes irrelevantes [Udu 91] [Buc 95] [Mon 94a];
2. Filtragem. Com operações de filtragem é possível suavizar (eliminação de bordas, por exemplo) ou realçar (extração de bordas, por exemplo) uma imagem. A idéia fundamental da filtragem é associar um valor de densidade a um voxel v na cena de saída, com base nas densidades de voxels numa pequena vizinhança de v na cena de entrada. Dependendo de como os valores de densidade são usados e da definição da vizinhança, obtêm-se diferentes operações de filtragem [Udu 91] [Gon 87] [Pra 91]. As operações de suavização receberam a designação de filtragem passa-baixa, e as de realçamento de filtragem passa-alta;
3. Interpolação de Imagens. É a operação de gerar fatias intermediárias em um volume amostrado. As cenas geradas pelos dispositivos de aquisição (*scanners*) normalmente não representam uma amostragem isotrópica de uma dada região, ou seja, as distâncias na direção z são maiores que as dimensões em x e y . Em algumas situações, a separação não uniforme entre fatias sucessivas é uma característica da própria aplicação. Para melhoria da precisão e, às vezes, da eficiência no processamento, é necessário gerar uma cena

- uniformemente distanciada a partir de uma amostragem não uniforme [Udu 91] [Pra 96];
4. Segmentação. Processo que subdivide uma imagem em suas partes constituintes ou objetos [Gon 87], possibilitando identificar estruturas de interesse.
 5. Mascaramento. O propósito do mascaramento é complementar a segmentação. Frequentemente, a segmentação automática falha e a única alternativa é um especialista da área de aplicação delimitar uma região sobre as fatias da cena de entrada, indicando onde está a estrutura de interesse. No pior caso pode ser necessário desenhar precisamente a fronteira da estrutura em todas as fatias da cena. Normalmente, entretanto, é suficiente especificar a região, chamada de região de máscara (*mask region*), tal que uma segmentação automática confinada a essa região possa extrair a estrutura de interesse [Udu 91].

Além das operações listadas acima, duas são de interesse especial para o projeto de reconstrução de um dente virtual. Elas são chamadas de registro e restauração e são apresentadas na próxima Seção.

2.2 Registro (*Registration*) e Restauração (*Restoration*)

Para a obtenção das várias fatias de um volume de dados pode ser necessário fatiar fisicamente o material original. Esse processo de fatiamento pode produzir distorções nas fatias individuais e, conseqüentemente, nas imagens geradas [Mon 94a], amplificadas posteriormente pelo procedimento de digitalização. O processo de **registro** tem como objetivo possibilitar o alinhamento das imagens correspondentes às fatias de um volume de dados.

O registro de imagens pode ser guiado por características intrínsecas (estruturas bem definidas e localizadas) ao objeto a ser reconstruído, ou por marcas artificiais inseridas nele antes do fatiamento. Com base nesses marcadores de referência, são realizadas operações de translação e rotação para recuperar a imagem original [Mon 94a] [Doy 93].

As operações de translação e rotação fazem parte de um grupo maior de técnicas para recuperar a imagem original. Tais técnicas compõem a Restauração de Imagens. Para se recuperar a imagem original deve ser aplicado o processo inverso ao que causou a degradação [Gon 87]. De acordo com [Gon 87], existem dois tipos de operações básicas para restauração: **transformação espacial** (*spatial transformation*) e **interpolação de tons de cinza** (*gray-level interpolation*).

No caso de distorções geométricas, ou seja, um deslocamento em pontos correspondentes num par de imagens, podem ser aplicadas operações de transformação geométrica. Transformações espaciais produzem um rearranjo dos pontos na imagem. Esse rearranjo pode ser executado usando-se um subconjunto de pontos, chamados de pontos de ligação (*tiepoints*), de localização

bem conhecida (figura 2.3) [Gon 87]. Em geral, são utilizados vários desses subconjuntos para cobrir a imagem toda.

Figura 2.3 Pontos de ligação correspondentes em dois segmentos de imagem [Gon 87].
Baseado nesses pontos é estabelecido um modelo para a transformação de cada pixel da imagem degradada.

É possível que nas transformações espaciais sejam produzidas coordenadas não inteiras, que não têm sentido para imagens digitais. Nesse caso, é preciso determinar qual tom de cinza deve ser assinalado ao ponto original. A regra mais simples é buscar o tom de cinza associado ao ponto mais próximo ao ponto de coordenadas não inteiras (figura 2.4) [Gon 87].

Figura 2.4 Esquema da interpolação de tons de cinza [Gon 87].

Um tipo de transformação espacial bastante interessante é a transformação isogonal, que considera distorções de translação, rotação e escala. Essa transformação é descrita com mais detalhes na próxima subseção.

2.2.1 Transformação Isogonal

Na área de Fotogrametria, pertencente às Ciências Cartográficas, são estudados os processos de obtenção de medidas confiáveis pelo uso de fotografias e imagens, utilizando, principalmente, transformações entre sistemas de coordenadas [Mer 79].

Na confecção de uma cópia impressa a partir de um negativo fotográfico, vários fatores podem ser causa de distorções nessa cópia. Deformação do negativo durante sua manipulação e distorção causada pelas lentes da câmera são exemplos desses fatores. Parâmetros que caracterizam as distorções (translação, rotação e escala) podem ser estimados, possibilitando, assim, a aplicação de uma transformação inversa para obtenção de medidas mais confiáveis.

A escolha de um modelo de transformação a ser aplicado segue critérios cuja explicação foge ao escopo do presente texto. Nesta subseção é apresentada uma transformação denominada isogonal ou de similaridade, que considera mudança de posição (translação), de direção (rotação) e de escala uniforme nas direções x e y, sem variação de forma [Tom 84].

O modelo matemático da transformação isogonal é dado por [Tom 84]:

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \lambda \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x_c \\ y_c \end{bmatrix} + \begin{bmatrix} dx \\ dy \end{bmatrix}$$

(x_d, y_d) : posição devido à deformação;

(x_c, y_c) : posição correta;

λ : fator de escala nas direções x e y;

θ : rotação;

dx, dy : valores de translação.

A expressão pode ser escrita na forma de coordenadas homogêneas,

$$\begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} = \begin{bmatrix} \lambda \cos \theta & \lambda \sin \theta & dx \\ -\lambda \sin \theta & \lambda \cos \theta & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix}$$

mais comum em Computação Gráfica.

A determinação dos parâmetros pressupõe o conhecimento prévio das posições, correta e deformada, de alguns pontos. Para o desenvolvimento do modelo será utilizado uma forma mais simplificada,

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \begin{bmatrix} a & b \\ -b & a \end{bmatrix} \begin{bmatrix} x_c \\ y_c \end{bmatrix} + \begin{bmatrix} dx \\ dy \end{bmatrix}$$

onde $a = \lambda \cos \theta$ e $b = \lambda \sin \theta$.

Para cada ponto tem-se

$$x_d^i = a * x_c^i + b * y_c^i + 1 * dx + 0 * dy$$

$$y_d^i = -b * x_c^i + a * y_c^i + 0 * dx + 1 * dy.$$

Ou seja, cada ponto observado dá origem a duas equações. Como são procuradas quatro incógnitas, são necessários, no mínimo, dois pontos para resolver o sistema. Entretanto, um número maior de pontos produz resultados mais confiáveis.

Agrupando os pontos conhecidos e utilizando a forma matricial, o sistema pode ser escrito na forma a seguir.

$$\begin{bmatrix} x_d^1 \\ y_d^1 \\ \vdots \\ x_d^j \\ y_d^j \\ \vdots \end{bmatrix} = \begin{bmatrix} x_c^1 & y_c^1 & 1 & 0 \\ y_c^1 & -x_c^1 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_c^j & y_c^j & 1 & 0 \\ y_c^j & -x_c^j & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} a \\ b \\ dx \\ dy \end{bmatrix}$$

Ou, simbolicamente,

$$D = CX$$

onde: D é o vetor de pontos distorcidos (deformados);

C é a matriz com os pontos corretos;

X é o vetor de parâmetros.

X é encontrado pelo desenvolvimento da expressão anterior:

$$C^T D = C^T C X \Rightarrow [C^T C]^{-1} [C^T D] = [C^T C]^{-1} [C^T C] X \Rightarrow X = [C^T C]^{-1} [C^T D]$$

De posse dos parâmetros, é possível aplicar a transformação inversa à que provocou as distorções, dada por:

$$\begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} a & b \\ -b & a \end{bmatrix}^{-1} \begin{bmatrix} x_d - dx \\ y_d - dy \end{bmatrix} \Rightarrow \begin{bmatrix} x_c \\ y_c \end{bmatrix} = \frac{1}{a^2 + b^2} \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \begin{bmatrix} x_d - dx \\ y_d - dy \end{bmatrix}$$

Com a transformação inversa, o posicionamento correto dos demais pontos distorcidos pode ser obtido.

O conjunto de operações de pré-processamento é muito mais amplo do que o apresentado aqui. Neste capítulo pretendeu-se apresentar noções e aplicações de alguns tipos de pré-processamento, detendo-se mais detalhadamente naqueles que são de interesse ao projeto do dente virtual. Após a aplicação dessas operações na preparação dos dados, eles estão prontos para serem tratados pelos algoritmos de reconstrução, que é o assunto do próximo capítulo.

3. RECONSTRUÇÃO DE MODELOS 3D

Visualizar o corpo humano e suas estruturas internas é uma meta que tem desafiado artistas e cientistas através dos séculos. Por 500 anos, desde Leonardo da Vinci, desenhos têm sido o principal recurso para ensinar anatomia, porque eles permitem a mistura de realismo e abstração adequados aos propósitos didáticos. A descoberta do Raio-X permitiu observar o interior de um corpo vivo em chapas bidimensionais. Mais recentemente, a partir da década de 70, a Tomografia Computadorizada e o Imageamento por Ressonância Magnética possibilitaram a aquisição de dados em três dimensões. Com base nessas técnicas, a computação gráfica gerou os primeiros modelos 3D de um corpo vivo [Tre 96].

Os dispositivos de Tomografia Computadorizada e Imageamento por Ressonância Magnética geram um conjunto de imagens 2D, cada uma delas representando uma fatia (seção) do objeto varrido (*scanned*). Fatiar fisicamente um objeto e obter a imagem de cada seção também é uma outra possibilidade de obtenção de um volume de dados. Essas imagens podem ser obtidas diretamente no formato digital ou pela digitalização de fotos. Um objeto, ou parte dele, pode ser representado na forma tridimensional através de um processo de reconstrução, a partir de seu respectivo conjunto de imagens bidimensionais.

Existem duas abordagens básicas para a visualização de estruturas tridimensionais ou Visualização Volumétrica: *Rendering* por Superfícies e *Rendering* Volumétrico Direto. Nessa última é calculada a contribuição de cada voxel para o pixel associado na imagem final. Na primeira abordagem é extraída inicialmente a fronteira que forma a superfície do objeto a visualizar, gerando um modelo geométrico tridimensional que sofrerá, posteriormente, um processo de *rendering*.

Em [Udu 91] é apresentada uma classificação para as abordagens de formação de superfícies: agrupamento de contornos (*contouring*), tecelagem (*patching*), e traçado de superfície (*surface tracking*). Em agrupamento de contornos, a idéia é calcular a fronteira da estrutura em cada fatia com linhas de contorno e tratar a fronteira 3D da estrutura como uma pilha dessas linhas de contorno. Em tecelagem, a fronteira 3D é estimada como um conjunto de elementos de um formato básico (*patch*), normalmente triângulos, através da tecelagem das linhas de contorno de duas fatias sucessivas com tais elementos. Em traçado de superfície, a fronteira 3D é determinada através de um conjunto daquelas faces dos voxels que delimitam a estrutura. As linhas de contorno podem ser entendidas como um conjunto de pontos ($P_1, P_2, \dots, P_n$) que estão na fronteira.

Neste capítulo são apresentados métodos empregados no processo de reconstrução ou criação de modelos 3D. Inicialmente, são feitas considerações gerais sobre a reconstrução de superfícies e, em seguida, algumas abordagens são expostas e a abordagem de interesse no presente projeto é detalhada.

3.1 Aspectos da Triangulação entre Contornos

Existe uma grande variedade de aplicações nas quais os dados são coletados como um conjunto de seções ou fatias do objeto em estudo. Dessas fatias são extraídos os contornos de estruturas de interesse (figura 3.1). A superfície original de uma estrutura pode ser reconstruída através da conexão dos seus respectivos contornos, processados em pares de fatias consecutivas. A conexão ou a ligação entre contornos é executada normalmente utilizando-se retalhos triangulares, produzindo uma malha triangular representativa da superfície. Na área médica, por exemplo, essas técnicas são úteis na representação tridimensional de estruturas internas da anatomia humana (ossos, tumores, tecidos) [Eko 91] [Jon 94] [Oli 96].

Figura 3.1 Contorno de uma estrutura em uma fatia.

De acordo com as definições estabelecidas em [Jon 94], um contorno é um conjunto de segmentos de reta (bordas), cada um delimitado por dois vértices. Com relação a algumas propriedades geométricas, um contorno pode ser:

1. Fechado: Cada um dos seus vértices pertence a exatamente dois segmentos;

2. Simples: Não existe auto-interseção;
3. Convexo: Ocorrem, no máximo, duas interseções com uma linha reta arbitrária;
4. Ordenado: O contorno é fechado e os segmentos estão organizados de tal forma que eles são colocados fim-a-fim, juntando o vértice do último segmento com o do primeiro;
5. Orientado: O contorno é fechado, simples, e tem seus segmentos organizados de tal forma que um ponto está dentro do contorno se está a direita dele; caso contrário, está fora.

A solução do problema geral de triangulação é encontrar uma superfície poliedral a partir de um conjunto de vértices arbitrariamente distribuídos no espaço. Entretanto, para o caso de triangulação entre contornos em fatias num volume de dados, é importante explorar o fato dos contornos estarem localizados em planos bem definidos. Isso reduz esse problema a uma triangulação entre contornos em duas fatias consecutivas [Eko 91].

Os problemas identificados para conectar contornos em duas fatias consecutivas, que devem ser tratados pelos métodos, são os descritos a seguir:

1. Problema de correspondência (*correspondence problem*) [Jon 94]: Determinação dos pontos, presentes em fatias consecutivas, que estão relacionados, ou seja, que devem ser conectados. Sem o estabelecimento da correspondência não é possível realizar a tecelagem dos contornos;
2. Problema de tecelagem e ramificação (*tiling and branching problem*) [Oli 96]: Estabelecimento de como os contornos de um par de fatias consecutivas devem ser conectados, após estabelecidas as correspondências. Alguns métodos requerem a interação com o usuário para dirigir o processo de tecelagem [Jon 94]. Várias situações podem ocorrer:
 - a. Nenhuma conexão: Caso degenerado, no qual um contorno em uma fatia não tem correspondente na outra fatia (figura 3.2(a));
 - b. Tecelagem simples (*simple tiling*): Situação na qual existe correspondência um-para-um, isto é, um contorno na primeira fatia corresponde a um único na segunda fatia (figura 3.2(b)). Em [Eko 91], esse caso é denominado ramificação única (*single-branching*);
 - c. Ramificação simples (*simple branching*): Quando existe correspondência de um contorno com dois ou mais na outra fatia (figura 3.2(c)). Em [Eko 91], a situação é definida pelo termo ramificação múltipla (*multiple branching*);
 - d. Multi-ramificação (*multi-branching*): Caso no qual vários contornos de uma fatia correspondem a vários contornos na outra fatia (figura 3.2(d)). A figura 3.2(d) mostra também a ocorrência de buracos.
3. Problema de ajuste de superfície (*surface fitting problem*) [Oli 96]: Suavização através de uma superfície “ótima” sobre a malha calculada previamente. A própria malha pode ser considerada como a superfície na maioria dos casos.

Figura 3.2 Problemas de tecelagem e ramificação [Oli 96].
(a) Nenhuma conexão. (b) Tecelagem simples. (c) Ramificação simples.
(d) Multi-ramificação (exemplo com buraco, a direita).

Uma solução para o problema de correspondência é considerar que contornos relacionados se sobrepõem e verificar o grau dessa sobreposição [Eko 91] [Jon 94] [Oli 96]. O problema de ramificação pode ser resolvido pela criação de fatias interpoladas intermediárias às duas originais. Essas fatias servem de auxiliares no processo de tecelagem [Eko 91] [Oli 96]. Os métodos para a conexão de contornos em fatias de um volume de dados têm evoluído, procurando oferecer soluções cada vez mais genéricas, que englobem as diversas situações decorrentes dos problemas citados.

As soluções para o problema de triangulação entre contornos podem ser divididas em dois grupos: ótima e heurística. Os métodos ótimos buscam a melhor triangulação de acordo com algum critério. É utilizado um grafo para descrever os possíveis arranjos de triangulação. Cada vértice do grafo representa um par de pontos conectados e cada arco corresponde a um possível retalho triangular. É definida uma função custo para cada arco, que é utilizada na determinação do melhor caminho de acordo com o critério estabelecido (máximo volume ou mínima área, por exemplo). Os métodos heurísticos são computacionalmente mais baratos e consistem em definir triângulos um a um utilizando critérios locais de decisão, como por exemplo, escolher a borda mais curta para formar o triângulo. Em situações nas quais os contornos são bastante diferentes na forma, orientação ou posição, ou apresentam distorção, os algoritmos heurísticos podem produzir resultados incorretos [Eko 91].

Além dos métodos baseados em triangulação de contornos vizinhos, pode-se reconstruir estruturas com base em algoritmos de traçado de superfícies de nível, aplicados isoladamente ou em conjunto com estratégias geométricas [Shi 98a,98b]. Na próxima Seção, são vistos três métodos de geração de modelos 3D.

3.2 Métodos de Reconstrução de Superfícies

No desenvolvimento de novas abordagens, procura-se resolver questões não tratadas pelas antecessoras ou melhorar a eficiência e a consistência de soluções de pontos já estudados

anteriormente, ou ainda adequar uma abordagem aos requisitos de uma aplicação específica. A seguir estão listados três métodos e delineadas suas características gerais:

1. **Triangulação a partir de múltiplos contornos com formato arbitrário:** Em [Eko 91], é definido um procedimento geral de triangulação que provê soluções para os problemas de obtenção de resultados incorretos quando os contornos não são convexos ou têm formas muito diferentes, e para o problema de ambiguidade. Para a solução do primeiro problema considera-se que os contornos têm o seu fecho convexo¹ (*convex hull*) similares em forma e orientação. Para resolver o caso de ambiguidade supõe-se que a amostragem na direção z é suficiente para o estabelecimento das correspondências, isto é, contornos relacionados apresentam um certo grau de sobreposição. Esse método foi aplicado na reconstrução de uma vértebra humana, incluída em resina e fatiada em 24 pedaços, seguida de um procedimento de digitalização usando uma câmera de vídeo;
2. **Construção de superfícies utilizando funções de campo:** No trabalho realizado por Jones e Chen [Jon 94] é apresentada uma abordagem que pode manipular condições ambíguas sem empregar um algoritmo de estabelecimento de correspondência entre contornos. Seu algoritmo requer que os contornos, com relação às propriedades geométricas, sejam fechados e simples. O algoritmo é composto por dois passos básicos: construção de volume e construção de superfície. No primeiro passo, é gerada uma malha a partir de um conjunto de fatias e de uma função de campo (*field function*). Essa representação é utilizada, no segundo passo, para extrair a superfície de interesse. Esse algoritmo foi testado na reconstrução de tronco (17 fatias) e pulmões (12 fatias), com dados obtidos por MRI;
3. **Reconstrução 3D por extração e divisão de áreas de diferença:** Em [Oli 96] é descrito um método que, segundo os autores, permite construir, de forma totalmente automática, uma superfície com uma topologia válida mesmo quando ocorrem severas variações topológicas e morfológicas. Primeiro, é estabelecida uma correspondência global entre cada par de fatias adjacentes através da extração de áreas de diferença (*areas of difference*). Em seguida, cada área de diferença é particionada por uma rede de bissetores angulares (ABN - *Angular Bisector Network*). Essa rede é utilizada para guiar um processo de triangulação e para auxiliar na geração de contornos interpolados, quando uma triangulação direta não for possível, em fatias intermediárias. A triangulação final, para cada par de fatias, é obtida pela concatenação das triangulações nas áreas de diferença. O método foi aplicado na reconstrução de uma pélvis humana a partir de uma descrição de 23 fatias.

As abordagens 1 e 3 acima tentam reconstruir a superfície resolvendo a geometria entre fatias vizinhas e o relacionamento entre contornos vizinhos, formando um modelo 3D pela triangulação do espaço entre contornos de fatias adjacentes [Shi 98a,98b]. Uma outra abordagem para esse

¹ Um fecho convexo de um contorno é o menor polígono convexo no qual esse contorno está contido [Pre 85].

problema recentemente elaborada é apresentada em [Non 98], que procura resolver os problemas de ambiguidade, e ao mesmo tempo eliminar instabilidades causadas por outros métodos.

Dentre as abordagens apresentadas acima, a segunda se mostrou adequada à resolução do problema do projeto de reconstrução tridimensional de um dente (detalhes no capítulo 5). Na Seção seguinte, essa abordagem é apresentada com mais detalhes.

3.3 Construção de Superfícies Utilizando Funções de Campo

Muitos métodos de reconstrução apresentam um passo de estabelecimento de correspondência entre contornos para resolver casos de ambiguidade. Esses métodos apresentam dois inconvenientes principais:

1. Eles falham em manipular condições ambíguas ou requerem algoritmos complicados para lidar com o problema, muitas vezes resultando em inconsistência e ineficiência;
2. Eles manipulam somente contornos com propriedades similares, tais como orientação, convexidade e forma.

Em [Jon 94] é apresentada uma abordagem para a construção de superfícies, a partir de um conjunto de fatias, que não requer a utilização de algoritmos de estabelecimento de correspondência entre contornos para manipular os casos de ambiguidade. Com relação a propriedades geométricas, os contornos devem ser fechados e simples. O algoritmo de construção de superfícies procura atender aos seguintes critérios:

1. Precisão. É produzida uma superfície a partir da qual os contornos originais podem ser reproduzidos quando ela é interceptada pelos planos correspondentes às fatias originais;
2. Consistência. O formato da superfície não é afetado pela ordem em que as fatias e os segmentos são processados;
3. Eficiência. Apresenta facilidade de implementação e rapidez de execução;
4. Flexibilidade. Requer a satisfação, pelos contornos, de poucas propriedades geométricas (fechado, simples, convexo, ordenado e orientado).

O algoritmo é composto de duas fases principais: **construção de volume** e **construção de superfície**. Os dados de entrada para a primeira fase são o conjunto de fatias e uma função de campo, que é usada para verificar se um ponto está fora, sobre, ou dentro de um contorno. Nessa fase é produzida uma representação do volume, a qual serve de dado de entrada para a segunda fase (figura 3.3).

Figura 3.3 As fases do algoritmo apresentado em [Jon 94].

Na fase de construção de volume, uma grade bidimensional é mapeada sobre cada fatia do conjunto inicial (ver figura 3.4). Dessa forma, cada ponto da grade pode ser considerado como o vértice de uma célula do volume, de coordenadas (x,y,z) , cujo valor associado é dado pela função de campo $f(x,y)$, calculada sobre todos os contornos do respectivo plano z . Melhor precisão é obtida pela definição de uma grade fina ou pela escolha de uma grade na qual cada vértice de contorno esteja situado num vértice da grade. A extração de contornos de uma fatia é feita pela busca de pontos onde $f(x,y) = 0$. Considerando que as bordas são segmentos de reta, é possível aplicar uma interpolação linear para determinar o valor de um ponto que não pertença à grade (um segmento de reta é uma interpolação linear entre dois vértices). Na prática, por questões de eficiência, pode-se utilizar uma grade mais grosseira juntamente com uma função de campo apropriada.

Figura 3.4 Grades bidimensionais formando um volume.

Seguindo o fluxo da figura 3.3, na fase de construção de superfície, a representação de volume produzida é tratada como um espaço contínuo, no qual um ponto fora da grade, ou seja, que não está numa célula, tem seu valor determinado pela interpolação trilinear dos valores dos seus oito vizinhos. Os pontos nos quais $f(x,y,z) = 0$ identificam os pontos que pertencem à superfície delimitada pelos contornos do conjunto de fatias. Esse volume pode ser usado tanto para um *rendering* por superfície como para *rendering* volumétrico direto.

3.3.1 Aplicação da Função de Campo

A função de campo $f(x,y)$ é definida sobre todos os pontos de um plano no qual estão os contornos de uma fatia, de forma que

$$f(x,y) \begin{cases} < 0, & (x,y) \notin \text{a nenhum contorno;} \\ = 0, & (x,y) \in \text{a um contorno;} \\ > 0, & (x,y) \in \text{ao interior de um contorno.} \end{cases}$$

Uma função de campo simples é uma função de decisão que determina se um ponto está fora, dentro ou sobre um contorno da forma descrita a seguir:

$$f(x,y) \begin{cases} -1, & (x,y) \notin \text{a nenhum contorno;} \\ 0, & (x,y) \in \text{a um contorno;} \\ 1, & (x,y) \in \text{ao interior de um contorno.} \end{cases} \quad (\text{eq. 3.1})$$

Com a definição acima é possível utilizar um algoritmo de preenchimento de polígono para determinar os valores dos pontos de uma grade em cada fatia. Entretanto, a precisão da superfície produzida é excessivamente afetada pela resolução utilizada para construir a grade. Na figura 3.5(b), são apresentados os diferentes resultados obtidos, com três grades de resoluções diferentes, para a porção mostrada na figura 3.5(a).

Figura 3.5 Resultados do uso de diferentes resoluções e funções de campo [Jon 94].

Uma outra possível função de campo é uma função de distância, definida como

$$f(x,y) \begin{cases} -dist(x,y), & (x,y) \notin \text{a nenhum contorno}; \\ 0, & (x,y) \in \text{a um contorno}; \\ dist(x,y), & (x,y) \in \text{ao interior de um contorno}. \end{cases}$$

onde $dist(x,y)$ é a distância do ponto (x,y) ao contorno(s) próximo(s). O resultado do uso dessa função é um volume que representa uma função contínua amostrada sobre uma grade discreta e que permite a construção de uma superfície pela aplicação de uma interpolação linear. Ainda, a superfície gerada não é influenciada, como no caso de uso da função de campo anterior, pela resolução da grade (figura 3.5(c)). A figura 3.6 mostra as diferentes superfícies geradas pelo uso de cada uma das funções descritas.

Figura 3.6 Superfícies geradas por diferentes funções de campo [Jon 94].
Na reconstrução à esquerda utilizou-se função de decisão, na direita, função de distância.

A aplicação direta da função de distância, calculando o seu valor para todos os pontos do volume, pode apresentar um custo computacional alto, uma vez que para cada ponto da grade, numa fatia, é necessário o cálculo de suas distâncias relativas a cada contorno daquela fatia, a fim de encontrar a distância mínima. Tomando um exemplo numérico, usando uma grade de 200×200 pontos, num volume composto de 25 fatias, com uma média de 100 segmentos de contorno por fatia, é preciso calcular $200 \times 200 \times 25 \times 100 = 100$ milhões de distâncias. A complexidade desse tipo de algoritmo pode ser reduzida através de um particionamento em cada fatia. Cada fatia é particionada em uma quantidade de setores igual ao número de segmentos presentes na fatia. Cada setor está associado a um segmento e é construído de tal forma que os pontos nele contidos estão mais próximos a esse segmento do que a qualquer outro segmento na fatia (figura 3.7). Assim, o cálculo de distância é realizado em relação a um único segmento.

Figura 3.7 Particionamento de uma fatia em setores [Jon 94].

Apesar da redução da quantidade de cálculo, ainda assim é necessário fazer cálculos para todos os pontos da grade. Além disso, a implementação do algoritmo de particionamento é complexa. Por esses motivos, a redução obtida no custo computacional pode ser pequena. Uma outra solução proposta baseia-se na observação de não ser necessário o cálculo preciso das distância em todos os pontos. Para extrair uma superfície, um algoritmo como *Marching Cubes* [Lor 87], que possui um critério local de determinação de interseção, examina cada célula de um volume verificando se os seus vértices (pontos da grade) têm sinais diferentes (negativo, zero ou positivo), ou seja, têm posições diferentes (fora, sobre, dentro) relativas à superfície. Para as células nessas condições, é aplicada uma interpolação linear para realizar a construção da superfície internamente à célula (figura 3.8). Para a abordagem de reconstrução, os vértices dessas células são os pontos que estão próximos à superfície real e, portanto, precisam ter os valores de distância calculados com precisão. Assim, a quantidade de cálculos de distância fica reduzida aos casos que “passam” por esse critério de interseção. O algoritmo *Marching Cubes* é melhor detalhado no capítulo 4.

Figura 3.8 Exemplo de interpolação no *Marching Cubes* [Jon 94].

Para implementar a solução descrita no parágrafo anterior, cada ponto da grade tem associado dois campos: estado e distância. O estado de um ponto (fora, sobre ou dentro do contorno) é calculado pela função de decisão (eq. 3.1) utilizada em conjunto com um procedimento de varredura (*scan-converting*) para os contornos em uma fatia. Para um ponto (x,y,z) de um volume, não é necessário fazer o cálculo da distância nos seguintes casos:

1. Se o valor do campo estado é zero, ou (ponto sobre um contorno)
2. Se
$$\sum_{k=z-1}^{z+1} \sum_{j=y-1}^{y+1} \sum_{i=x-1}^{x+1} \text{estado}(i,j,k) = 27 \text{ ou } -27$$
 (pontos estão todos dentro ou fora)

Com isso, apenas aqueles pontos que influenciam o processo de construção da superfície terão os valores do campo de distância calculados.

Uma abordagem alternativa para a reconstrução de superfícies é a que utiliza técnicas baseadas em modelos deformáveis [Alg 96]. Em tal abordagem, uma aproximação inicial para a superfície é

adotada. Esse modelo inicial é então “deformado” para chegar ao modelo final do objeto, empregando modelos dinâmicos que governam o ajuste da superfície. Modelos deformáveis têm sido estudados para tentar resolver o caso geral de reconstrução de uma superfície S , partir de um conjunto P de pontos não estruturado. Maiores detalhes podem ser vistos em [Shi 97].

Uma vez extraída a superfície, as etapas seguintes no processo de visualização compreendem a apresentação gráfica e a interação com o modelo. O próximo capítulo descreve essas etapas.

4. VISUALIZAÇÃO PARA MODELOS 3D

Visualização é definida como “transformação de conceitos abstratos em imagens reais ou mentalmente visíveis” (Novo Dicionário Básico da Língua Portuguesa Folha / Aurélio). De forma mais próxima à linguagem técnica da área de Ciências da Computação, pode-se definir visualização como a transformação de dados ou informação em imagem. A visualização explora as excelentes características do sentido da visão (estima-se que 50% dos neurônios estão associados com esse sentido [Wat 92]), bem como a capacidade de processamento da mente humana, buscando obter um meio efetivo e simples para exibir informações complexas e/ou volumosas. De forma complementar, é possível explorar os sentidos de audição (som) e tato (toque) [Sch 96, 97] [Min 95].

Várias classificações diferentes existem para descrever visualização. **Visualização Científica** (*Scientific Visualization*) é o nome formal dado à área da Ciência da Computação que abrange interface com o usuário e representação visual e de dados com estrutura geométrica implícita. Esse termo é restritivo, considerando que as técnicas de visualização são aplicadas em negócios, ciências sociais, demografia e gerenciamento de informações em geral. **Visualização de Dados** (*Data Visualization*), outro termo utilizado, é mais geral que o anterior, e implica o tratamento de dados oriundos de outras áreas além da de Ciências e Engenharia. Dessas outras áreas fazem parte a financeira e a de marketing, além de incluir, pela abrangência atribuída ao termo, aplicações de métodos estatísticos e técnicas-padrão de análise de dados. Mais recentemente, surgiu o termo **Visualização de Informação** (*Information Visualization*). Nesse campo, esforços são voltados para a visualização de informações abstratas, tais como documentos de hipertexto na WWW (*World Wide Web*), estrutura de diretórios/arquivos em um computador ou estruturas de dados abstratas [Sch 96, 97] ou seja, dados sem características geométricas pré-definidas. O

presente texto está relacionado com a Visualização Científica.

A origem da visualização como uma disciplina formal data de 1987, quando foi publicado o relatório “*Visualization in Scientific Computing*”¹, no qual é apresentado um sumário de aspectos políticos, econômicos, educacionais e tecnológicos de Visualização Científica como uma disciplina emergente. Aí, também, foi estabelecido o termo Visualização Científica. Esse relatório retrata os resultados de um *workshop* de dois dias sobre “Visualização em Computação Científica”, realizado em fevereiro de 1987, com participação de representantes do setor acadêmico, da indústria e de laboratórios de pesquisa do governo americano. No relatório foram apresentadas várias conclusões, dentre elas o crescente volume de dados a serem analisados e a importância de equipes multidisciplinares na realização dessa tarefa.

Desde então, muitas conferências específicas ficaram estabelecidas, como a IEEE *Visualization* e o EUROGRAPHICS *Workshop on Visualization*. Conferências mais gerais, como ACM SIGGRAPH e EUROGRAPHICS, dedicam grande parte dos seus programas à tecnologia de visualização. Essas conferências discutem assuntos como facilidade de dados (*Data Facilities*) e visualização volumétrica (*Volume Visualization*). O primeiro está relacionado com as facilidades de uso e transferência de informação, o segundo com a representação de informações 3D e a visualização do interior de objetos [Sch 96, 97] [Bro 92].

Processos físicos, tais como a trajetória de projéteis e o fluxo de fluidos, são exemplos de casos que têm sido simulados em computadores. O aumento na quantidade de dados gerados pelos cálculos nas simulações tornou necessária a transformação de números em imagens, devido à limitação humana em assimilar e entender tamanha quantidade de dados. Imagens têm o poder de apresentar grandes quantidades de informação numérica de forma a permitir uma compreensão maior dessa informação. O desenvolvimento da computação gráfica e a disponibilidade de equipamentos computacionais cada vez mais potentes têm provocado um aumento na importância da visualização como ferramenta para manipular a enchente de informações do mundo atual [Bro 95] [Sch 96, 97].

A Visualização está preocupada com a exploração de dados e informação graficamente, como um meio de ganhar entendimento e discernimento dos dados [Bro 95] [Sch 96, 97]. Para atingir esse fim, são empregados conhecimentos das áreas de computação gráfica, interface com o usuário, processamento de imagens e projeto de sistemas [Bro 92].

A Visualização, através das etapas de transformação, mapeamento e exploração, produz um

¹McCormick, B.; DeFanti, T. A.; Brown, M. D. - “*Visualization in Scientific Computing*”, ACM SIGGRAPH Computer Graphics, vol. 21, no. 6, November 1987.

conjunto de dados gráficos (primitivas gráficas), que são, então, convertidos para uma imagem. Nesse processo de conversão, denominado de *rendering*, a preocupação maior está centrada no poder da imagem de transmitir o conteúdo da informação, mais do que no seu fotorealismo. Isso, somado à capacidade de interação com os dados, formam um ferramental adequado à exploração e interpretação dos dados [Sch 96, 97] ou do fenômeno que os gerou. A interatividade é essencial para a visualização de estruturas multidimensionais, pois usuários precisam frequentemente modificar os dados e os gráficos para obter diferentes pontos de vista, e assim, ganhar uma compreensão adicional da complexidade inerente à estrutura sob investigação [Sen 90].

Os dados em Visualização Científica variam enormemente e, usualmente, são complexos e multidimensionais. Eles podem ser:

1. 2D, 3D ou multidimensionais;
2. Organizados em uma malha ou esparsos;
3. Do tipo escalar, vetorial ou tensor;
4. Estáticos ou variantes no tempo.

As técnicas para visualização variam de acordo com a dimensão e organização do conjunto de dados e dos aspectos que se deseja observar [Min 95,97]. Dentre as várias formas disponíveis, a visualização de estruturas tridimensionais, que recebe o nome de visualização volumétrica, tem recebido grande atenção. A seguir são apresentados tópicos de relevância relacionados com esse tipo de visualização e, na sequência, são descritos aspectos relativos à interação usuário-computador.

4.1 Visualização Volumétrica

Visualização volumétrica é o processo de projetar um conjunto de dados multidimensional sobre um plano bidimensional (imagem) com o propósito de obter um entendimento sobre a estrutura (ou falta dela) contida no volume de dados. Ela supre as seguintes necessidades:

1. Análise e entendimento de dados volumétricos obtidos em experimentos ou por varredura (principalmente 3D);
2. Síntese de objetos volumétricos através de um modelo computacional;
3. Transformação, manipulação e interação com os dados;
4. Produção de imagens a partir desses conjuntos de dados complexos.

Esse tipo de visualização foi criado inicialmente para visualizar dados escalares, e posteriormente, para vetores e tensores. A comunicação sobre o uso de dados volumétricos, particularmente imagens médicas 3D, data da metade da década de 70 [Elv 92] [Kau 90] [Sch 96, 97].

A animação é um fator crítico para um processo de visualização volumétrica. Sem a capacidade de animar uma visualização, o usuário encontra dificuldades em entender informações tridimensionais [Elv 92] principalmente as variantes no tempo. Como mencionado anteriormente, uma boa interface, que forneça condições apropriadas de interação, é bastante importante na exploração de dados em um sistema de visualização. Outras características desejáveis para as técnicas de visualização volumétrica, ainda segundo [Elv 92], são: representação de dados compreensível, rápida manipulação de dados e um processo de *rendering* veloz.

Serão abordados nesta Seção alguns algoritmos básicos, fundamentais no campo de visualização volumétrica. Nas próximas subseções são apresentados alguns conceitos e definições, importantes para o entendimento dos referidos algoritmos.

4.1.1 Uma Terminologia para Visualização Volumétrica

Apesar de não existir um consenso sobre a nomenclatura a ser usada para visualização volumétrica, a definição de alguns termos é interessante e útil para a descrição e entendimento das técnicas. A figura 4.1 apresenta, além desses termos, um processo geral de visualização volumétrica.

Figura 4.1 Terminologia para Visualização Volumétrica [Kau 90].

Os dados são gerados por dispositivos de amostragem ou por modelos computacionais, sendo produzida uma amostra dos pontos de um objeto real ou simulado, geralmente uma sequência de varredura de seções. Esse conjunto de dados pode ser processado e filtrado por técnicas de processamento de imagens e, então, ser reconstruído para um volume de dados pelo

empilhamento das seções e uma interpolação entre elas. Um objeto volumétrico é representado tipicamente por uma grade tridimensional de voxels conforme descrito no capítulo 3, com um ou mais valores numéricos associados a cada voxel. Esses valores representam propriedades do objeto que foram medidas dentro do voxel. Uma fonte de volume de dados alternativa é a conversão de um modelo geométrico contínuo em uma malha volumétrica. Esse processo de conversão é denominado voxelização. O passo inverso, gerar um modelo contínuo a partir de uma matriz de voxels, também é possível. Isso é feito por um processo de extração de superfície (veja capítulo 3). A visualização pode ser feita seguindo-se duas abordagens básicas. Na primeira abordagem, chamada de *Rendering* Volumétrico Direto, o volume de dados é exibido diretamente do volume discreto de voxels no plano discreto de pixels. Na outra abordagem, denominada *Rendering* por Superfície, a visualização é feita a partir da representação da superfície, extraída do volume de dados [Kau 90]. Os métodos de *rendering* por superfície são, normalmente, mais rápidos que os de *rendering* volumétrico, uma vez que o volume é percorrido apenas uma vez para extrair a superfície [Sch 96, 97] [Elv 92] [Kau 90].

No processo de *rendering*, alguns dados iniciais devem ser fornecidos para os algoritmos. Aqueles que operam por *rendering* por superfície precisam de um valor característico (*threshold*) para extrair a superfície desejada. No caso de algoritmos de *rendering* volumétrico direto, devem ser estabelecidas duas tabelas: uma de cores, que mapeia os valores dos elementos de um volume de dados para cores significativas, e uma outra de opacidade, que serve para controlar a exibição de estruturas de interesse e manter transparente aquelas que não são de interesse [Elv 92].

Exemplos concretos de fontes de volume de dados são as simulações que utilizam análise de elementos finitos e dinâmica de fluidos, e a aquisição de imagens por Tomografia Computadorizada e Ressonância Magnética. Apesar de gerados de formas diversas, esses volumes podem ser tratados de forma similar [Elv 92].

4.1.2 Voxels e Células

Um volume de dados pode ser tratado como uma matriz de voxels ou uma matriz de células. Essas duas abordagens surgem da necessidade de reamostragem do volume, utilizando uma interpolação entre pontos da grade durante o processo de *rendering*. Entretanto, não há informação que permita a verificação da confiabilidade da interpolação realizada. Na **abordagem de voxels**, cada voxel é considerado um hexaedro no qual os valores internos tem o mesmo valor do ponto correspondente da grade, ou seja, os valores dos pontos de uma área que circundam um ponto da grade apresentam um valor invariante. Essa abordagem tem a vantagem de não fazer nenhuma suposição sobre o comportamento dos dados. Na **abordagem de células**, o volume é considerado como um conjunto de hexaedros cujos vértices são os pontos da grade, e o valor dos

pontos entre os vértices varia. Tal variação é estimada comumente por funções de interpolação linear ou tricúbica. As imagens geradas utilizando essa última abordagem são mais suaves comparadas com as da outra abordagem [Elv 92]. A figura 4.2 ilustra a diferença entre voxel e célula.

Figura 4.2 Exemplo de voxel (esquerda) e célula (direita).
O ponto em negrito é um ponto da grade.

4.1.3 Formas de Criação da Imagem

As imagens podem ser criadas de duas formas distintas: baseada no espaço da imagem (*image-order*) ou baseada no espaço do objeto (*object-order*). Espaço da imagem está relacionado com os elementos que compõem uma imagem, ou seja, os pixels, enquanto espaço do objeto está ligado com os elementos do volume de dados. Na primeira forma, *image-order*, o espaço da imagem é percorrido pixel a pixel, identificando quais e como os elementos de um volume afetam um determinado pixel. Na forma *object-order*, cada elemento do volume é projetado sobre o plano da imagem e é calculada a sua contribuição para cada pixel. Nessa última forma, o volume pode ser processado numa ordem normal, de frente-para-trás (*front-to-back*), ou inversa, de trás-para-frente (*back-to-front*). A vantagem do processamento frente-para-trás é que, assim que os elementos da frente gerem uma imagem completamente opaca, os elementos de trás podem ser desconsiderados. O outro tipo, trás-para-frente, apresenta a vantagem de permitir que o usuário veja os objetos do fundo durante a formação da imagem [Elv 92] [Sch 96, 97].

A seguir são apresentados alguns algoritmos de visualização volumétrica bastante conhecidos. *Contour-connecting*, *Opaque Cube*, *Marching Cubes*, *Marching Tetrahedra*, *Dividing Cubes* e *Ray-casting*, dos quais apenas o último corresponde a *rendering* volumétrico.

4.1.4 Contour-connecting

Foi um dos primeiros métodos definidos para gerar imagens em visualização volumétrica. É um método de *rendering* por superfície baseada no espaço do objeto, que, no primeiro passo, processa cada fatia individualmente procurando encontrar contornos fechados que estejam de acordo com um valor característico fornecido pelo usuário. Originalmente, a busca de contornos era feita manualmente; atualmente existem técnicas de processamento de imagens que fazem isso

automaticamente. O segundo passo envolve resolver o problema de tecelagem, normalmente usando elementos triangulares, através da conexão de contornos em duas fatias adjacentes (figura 4.3) [Elv 92] [Owe 95]. No capítulo 3, esse grupo de métodos recebeu a denominação de tecelagem (*patching*).

Figura 4.3 Busca (a) e conexão de contornos (b) [Owe 95].

4.1.5 Opaque Cube ou Cuberille

Esse algoritmo é executado em dois passos. No primeiro, a partir do valor característico fornecido pelo usuário, são procuradas as células cujos valores dos vértices enquadrem o valor característico, ou seja, o valor de um vértice seja maior que o valor característico e o do seu vizinho seja menor. Para cada célula é gerado um conjunto de 6 polígonos, um para cada face. Essa representação, no passo seguinte, sofre um processo de *rendering*. Essa representação faz com que a superfície na imagem gerada tenha um aspecto bloqueado [Elv 92] [Owe 95]. Veja figura 4.4.

Figura 4.4 Representação de superfície por cubos [Owe 95].

4.1.6 Marching Cubes, Marching Tetrahedra e Dividing Cubes

A partir de um valor característico da superfície, fornecido pelo usuário, o algoritmo *Marching Cubes* procura por células que possam contribuir para a formação da superfície desejada. Um grupo de quatro fatias adjacentes é lido para a memória; os gradientes (usados para *shading*²) são calculados; e cada célula entre as duas fatias intermediárias é processada a fim de identificar aquelas cujos os valores dos vértices enquadram o valor característico. Esse procedimento se repete, para cada grupo de quatro fatias, até que o volume todo seja processado. Cada célula identificada é processada para verificar de que forma ela e a superfície se interceptam. Os vértices

² *Shading* é o processo de tonalizar um objeto [Sch 97].

da célula são numerados e, para cada vértice, é assinalado 1 ou 0, dependendo do seu valor associado estar acima do valor característico ou abaixo. A combinação desses 1's e 0's forma um número de um byte de comprimento, que serve de índice para uma tabela de interseção de bordas (figura 4.5) [Elv 92] [Owe 95] [Sch 96, 97].

Figura 4.5 Numeração dos vértices e formação do índice no *Marching Cubes* [Owe 95].

Essa tabela informa, para cada índice, quais bordas são interceptadas pela superfície. É feita uma interpolação para determinar o ponto de interseção e também o valor do gradiente. Considerando que cada borda é interceptada apenas uma vez, quatro triângulos são suficientes para indicar o caminho da superfície dentro da célula. Existem 256 formas de combinação de quatro ou menos triângulos numa célula; através de reflexão e rotação essas formas podem ser reduzidas a 15 casos (figura 4.6) [Elv 92] [Owe 95] [Sch 96, 97].

Figura 4.6 15 casos do *Marching Cubes* [Sch 97].

No algoritmo *Marching Cubes* é possível a ocorrência de uma conexão errada de pontos na geração dos triângulos, levando ao aparecimento de triângulos inexistentes e vice-versa. Uma solução para esse problema é a divisão da célula em 5, 6 ou 24 tetraedros, e fazer o teste de interseção nesses tetraedros, nos quais dois triângulos são suficientes para cobrir as suas 6 arestas. Esse algoritmo, denominado *Marching Tetrahedra*, gera mais triângulos que o anterior e, portanto, é computacionalmente (processamento e armazenamento) mais caro. A figura 4.7 mostra a divisão de uma célula em 5 tetraedros [Elv 92] [Owe 95].

Figura 4.7 Exemplo de divisão de uma célula em tetraedros [Owe 95].

Uma outra situação que pode ocorrer é que um triângulo, quando projetado, pode ter dimensões menores que a de um pixel. Um algoritmo, o *Dividing Cubes*, tira proveito dessa observação da forma descrita a seguir. As células identificadas, aquelas que contribuem para a formação da superfície, são projetadas no espaço da imagem (tela) para verificar se a sua projeção atinge uma área maior que um pixel. Em caso positivo, a célula é dividida em subcélulas, as quais são consideradas como pontos de superfície. No caso negativo, a célula toda é considerada como um ponto de superfície. Pontos de superfície têm valor e localização no espaço-objeto, e são utilizados em substituição aos triângulos. Nesse algoritmo não é produzida uma representação intermediária, o que provoca uma economia no tempo de processamento. A figura 4.8 mostra o resultado da divisão de uma célula em subcélulas [Elv 92] [Owe 95].

Figura 4.8 Divisão de uma célula em subcélulas [Owe 95].

4.1.7 Ray-casting

É um algoritmo de *rendering* volumétrico direto bastante utilizado para a produção de imagens de alta qualidade. Ele é um algoritmo baseado no espaço da imagem (*image-order*) que calcula, para cada pixel, a cor e a opacidade correspondentes. Esse cálculo é realizado a partir do disparo

de um raio, partindo do pixel em direção ao volume de dados. As cores e opacidades, para cada pixel, são determinadas pela soma das cores e opacidades encontradas ao longo da trajetória do raio correspondente ao pixel. Tal trajetória termina quando a opacidade calculada atinge o valor máximo ou quando o raio ultrapassa o limite do volume. O raio segue uma trajetória reta sem desvios, pois reflexões do mesmo não são consideradas; isso diferencia esse algoritmo do *ray-tracing*. Na figura 4.9 é mostrado o caminho seguido por um raio disparado [Elv 92] [Owe 95].

Figura 4.9 Raio disparado do plano da imagem em direção ao volume [Owe 95].

A tecnologia de visualização, que se beneficia da enorme capacidade do sistema visual humano para perceber e interpretar dados, é parte integrante do processo exploratório de análise e compreensão de dados, e portanto, precisa estar acoplada a técnicas de interação, que é o assunto abordado a seguir.

4.2 Interação

Um conjunto de dados é apresentado em vários formatos com o objetivo de ter um melhor entendimento de sua natureza, e reconhecer ou descobrir padrões nele, exigindo uma flexibilidade na sua representação e uma forma progressiva de manipulação [Sen 90]. Nessa Seção, pontos relacionados com o assunto interação usuário-computador, que são úteis e podem ser aproveitados em sistemas de visualização, são apresentados.

4.2.1 Utilização de Metáforas

O uso de metáforas é um artifício comum na interação com o usuário. No domínio da linguagem, metáforas são utilizadas para expressar um conceito em termos de outros mais familiares. Observando interfaces de sistemas computacionais, pode-se notar que os objetos apresentados na tela, as designações de comandos e as formas de interação guardam uma estreita relação com elementos familiares do mundo real. Um exemplo é dado pela Xerox, que na década de 80,

percebendo o potencial do uso de metáforas em sistemas de computação, projetou uma interface baseada em um escritório. Essa interface era composta de representações gráficas (ícones) de objetos reais utilizados em um escritório, como por exemplo, papéis e pastas. Esses objetos ficam distribuídos e organizados sobre uma escrivaninha, a *desktop*. A interface foi concebida de tal forma que ações com objetos reais (por exemplo, jogar papel no lixo) tivessem seus equivalentes. Com isso, o usuário pode entender mais facilmente a forma de interação com o computador. Para tornar mais intuitiva a interação, foi desenvolvido o mouse para aliar-se ao monitor de vídeo e teclado [Pre 94]. Esse foi o desenvolvimento da clássica interface aproveitada pela Apple no microcomputador Macintosh, seguido pelo mercado de workstations e, mais recentemente adotada na implantação do Windows 95, da Microsoft.

É comum que a metáfora de interface não seja totalmente coincidente com o mundo físico, ou seja, pode ser necessária a inclusão de elementos, importantes na operação do sistema, que não tenham correspondentes na realidade do usuário. Exemplos desses elementos são as barras de rolagem (*scroll bars*), menus e windows.

Aspectos, subjetivos e emocionais, devem ser considerados na escolha do tipo de metáfora a ser utilizado. Uma mesma metáfora pode produzir efeitos diferentes quando aplicada a grupos de usuários diferentes. Por exemplo, a metáfora usando objetos e ações comuns em um escritório, como a descrita anteriormente, pode ser natural para adultos, mas não para crianças [Pre 94].

O monitor de vídeo tradicional tem uma área de exibição bidimensional. Entretanto, existem técnicas que permitem que objetos mostrados nessa área sejam percebidos como tridimensionais. Esse assunto é abordado na próxima subseção.

4.2.2 Representação Tridimensional

A visualização de objetos 3D em superfícies 2D envolve uma transformação, na qual pontos de um objeto, localizados no espaço, são projetados em um plano bidimensional, chamado de plano de projeção. Genericamente, uma operação de projeção transforma pontos definidos em um sistema de coordenadas de dimensão n para um outro sistema com dimensão menor que n . Projeções, quando feitas sobre planos, são conhecidas como projeções geométricas planares. A projeção de um objeto 3D é definida por raios de projeção (projetores) que partem do centro de projeção, passam por cada ponto do objeto, e interceptam o plano de projeção. Se a distância entre o plano e o centro de projeção é infinita, então a projeção é classificada como paralela; se a distância for um número finito, a projeção é dita perspectiva (figura 4.10) [Fol 96].

Figura 4.10 Projeção perspectiva e paralela [Fol 96].

Vários aspectos podem ser explorados para que o usuário tenha uma percepção de espacialidade de objetos 3D projetados em plano 2D. Uma delas são as **noção monocular de profundidade** (*monocular depth cue*) que utiliza algumas regras para que objetos pareçam ser tridimensionais. São elas [Pre 94]:

1. Tamanho. Entre dois objetos idênticos na forma, o maior parece estar mais próximo que o menor;
2. Interposição. Se um objeto esconde parte de um outro objeto, o segundo é percebido como posicionado atrás do primeiro;
3. Contraste, luminosidade e brilho. Objetos com maior nitidez e mais distintos parecem estar mais próximos, e objetos mais apagados, escurecidos, parecem estar mais afastados;
4. Sombra. Sombras provocadas por um objeto oferecem algumas indicações da posição relativa de objetos;
5. Textura. Quando a distância aparente aumenta, a textura de uma superfície detalhada se torna menos granulada.

Para compor uma imagem tridimensional daquilo que é visto, o cérebro humano faz uma recriação a partir das imagens que recebe de cada olho. Essas imagens são ligeiramente diferentes em função da separação dos olhos. De forma equivalente, um modelo tridimensional pode ser reconstruído usando **estereoscopia** ou **paralaxe binocular** (*binocular parallax*). Nessa técnica, os olhos esquerdo e direito recebem imagens de um mesmo objeto obtidas de ângulos ligeiramente diferentes [Pre 94] [Fol 96].

Uma outra noção de profundidade que pode ser utilizada é a **paralaxe de movimento** (*motion parallax*). Objetos observados através da janela de um ônibus em movimento parecem mover-se a taxas diferentes. Aqueles que estão mais distantes parecem deslocar-se mais lentamente que os mais próximos, dando uma indicação de distância e profundidade. Em uma tela, esse efeito pode ser obtido pela movimentação do ponto de visão pelo qual a imagem é vista [Pre 94].

A noção de três dimensões pode ir além da oferecida pelos artifícios discutidos nessa subseção. O usuário pode ser levado a ter a impressão de estar imerso num ambiente virtual, criado através da realidade virtual, tema da próxima subseção.

4.2.3 Realidade Virtual (VR - *Virtual Reality*)

Realidade Virtual é uma forma de criar mundos artificiais através da integração de imagens, sons, sensações de posição e tato através do computador. Ela permite ao observador mover-se de fora para dentro de uma imagem e, fazer uma exploração diretamente no espaço da imagem, tendo uma sensação de imersão no ambiente criado. A exploração pode ser, por exemplo, um caminhar (*walkthrough*) através de um projeto de arquitetura. Computadores são utilizados para criar ambientes tridimensionais com objetos que fornecem uma forte impressão de presença espacial, e têm comportamentos e aparências programadas e adequadas à tarefa do ambiente virtual [Bow 95] [Ros 95] [Bro 95] [Pre 94].

Realidade Virtual é um termo normalmente usado para referir-se a estilos de interação que têm em comum os seguintes fatores:

1. Sensação de presença física direta. Indicações sensoriais são criadas pela tecnologia para dar ao usuário uma forte sensação subjetiva de presença física e experiência direta. Essas indicações podem ser visuais, auditivas, táteis, de força sobre o corpo, ou alguma combinação delas;
2. Indicações sensoriais em três dimensões. Informações em pelo menos um dos canais sensoriais (visão, audição, tato) são apresentadas em três dimensões;
3. Interação natural. Tipicamente, sistemas de realidade virtual permitem uma manipulação de objetos, criados pelo computador, de forma similar ao usado com objetos do mundo real: apanhar, girar, jogar e assim por diante.

No aparato tecnológico mencionado no item 1 acima, para fornecer as indicações sensoriais, os elementos mais comuns são uma ou mais luvas (fornecem a sensação de toque em objetos gerados pelo computador), e um par de superfícies de exibição (telas de TV de grande-ângulo minuatizadas) montadas num conjunto especial colocado na cabeça [Pre 94].

Uma das virtudes de um sistema de realidade virtual é a de fornecer uma sensação de imersão no mundo virtual. Entretanto, por questões de mercado³, a realidade virtual foi dividida em duas áreas: sistemas imersivos e sistemas *desktop*. Os primeiros usam periféricos como luvas e capacetes. Os segundos empregam uma única e grande tela, um mouse tridimensional e um teclado.

Sistemas de realidade virtual fazem o usuário estar imerso num ambiente gerado pelo

³ Alguns produtores de sistemas CAD com receio da realidade virtual destruir suas posições no mercado, renomearam sistemas um pouco mais que um CAD animado como "realidade virtual *desktop*". (Entrevista de Ray Kalawsky em [Pre 94]).

computador, totalmente sintético. Sistemas de **Realidade Aumentada** (*Augmented Reality*) permitem ao usuário interagir com objetos sintetizados combinados com objetos do mundo real. Uma câmara posicionada num capacete de realidade virtual captura imagens da cena real, sobre as quais são arranjados objetos sintetizados, e são mostrados conjuntamente numa tela. Os objetos na tela, reais e sintetizados, devem estar perfeitamente registrados e alinhados, para não causar, por exemplo, a impressão de um objeto gerado pelo computador estar flutuando no ambiente real capturado. Nesses sistemas, que alguns autores denominam de **Realidade Realçada** (*Enhanced Reality*), as técnicas de processamento de imagem têm um papel bastante importante. Uma aplicação desses sistemas é a visualização, no paciente, de dados médicos como, por exemplo ultra-som [Baj 95] [Bow 95] [Sta 96].

Uma ferramenta bastante difundida atualmente é a VRML (*Virtual Reality Modeling Language*), que é uma linguagem para descrever imagens tridimensionais. O usuário pode explorar a cena 3D exibida, definida em arquivo VRML, através de controles no painel de um *browser* apropriado, que é acoplado a um *browser* WWW (<http://www.whatis.com>).

Diversos fundamentos, conceitos e definições importantes para este trabalho foram apresentados ao longo deste e dos capítulos anteriores. No próximo capítulo são apresentados tópicos relativos à realização da parte prática deste trabalho.

5. RECONSTRUÇÃO DO DENTE VIRTUAL - UMA IMPLEMENTAÇÃO

O objetivo do presente projeto de mestrado é elaborar e implementar um algoritmo para integrar um processo de reconstrução de modelos tridimensionais, e construir um sistema de exibição e exploração de estruturas dentárias internas e externas. A geração do modelo 3D de um dente é executada a partir de um conjunto de polígonos que descrevem os contornos 2D das partes de interesse presentes em fatias paralelas. No caso dentário, uma possibilidade de obtenção dessas fatias é o fatiamento de dentes reais.

O conjunto de polígonos pode ser obtido através da extração de contornos em imagens, oriundas de fotografias ou vídeos digitalizados, ou obtidas de dispositivos de captura acoplados a um microscópio. Os contornos podem, ainda, ser gerados por programas próprios para sua edição.

Uma aplicação imediata de um esquema desse tipo é em atividades de ensino e em áreas nas quais seja útil a apresentação de propriedades estruturais de dentes e a ilustração de procedimentos ou tratamentos odontológicos (clínicas e consultórios, por exemplo). A médio prazo, o sistema pode evoluir para um arcabouço completo que inclua treinamento de cirurgias e outros procedimentos críticos, realizado através de um modelo virtual completo de dentes e boca.

O algoritmo elaborado para reconstrução foi escrito na linguagem C++ e implementado no contexto da biblioteca de visualização VTK (*Visualization Toolkit*) versão 1.3 [Sch 96]. Essa biblioteca, além de disponibilizar vários algoritmos de visualização, oferece outras vantagens, tais como a possibilidade de anexação de novas classes e de adaptação a novas aplicações, além de possuir um exportador para o formato VRML.

Além da implementação do algoritmo para reconstrução 3D, foram desenvolvidos programas complementares para a preparação e pré-processamento dos dados, para correção geométrica de contornos, e para facilitar a configuração dos parâmetros da reconstrução.

Neste capítulo são apresentados tópicos relacionados ao VTK (Seção 5.1), à estrutura do projeto (Seção 5.2), à elaboração do algoritmo (Seção 5.3), à lógica e detalhes do filtro (Seção 5.4), e aos programas complementares (Seção 5.5).

5.1 Características do VTK

O *Visualization Toolkit* (VTK) é composto por uma biblioteca de classes que possui diversas técnicas para visualização e computação gráfica implementadas. Seu código está escrito em C++, e foi projetado segundo a filosofia de orientação a objetos[Rum 91], tendo como base a biblioteca gráfica OpenGL. VTK está disponível para diversas plataformas, inclusive microcomputadores PC executando, no mínimo, o sistema operacional Windows 95/NT [Sch 96,97][Min 97].

Esse *toolkit* pode ser obtido pela Internet (<http://www.cs.rpi.edu/~martink>) ou em CD-ROM juntamente com livro escrito por seus criadores ([Sch 97]). O código fonte também é fornecido. Um aspecto adicional importante do VTK é o seu baixo custo.

Além da comunicação com a linguagem C++, as classes do VTK podem ser utilizadas a partir do Tcl/Tk¹ para a execução de visualizações. Complementando as características gráficas e de visualização, o VTK permite que o usuário possa interagir com a cena gerada, através de operações de transformação geométrica e zoom. Além disso, um objeto apresentado pode ser visto no formato de superfície ou de *wireframe* (fio de arame). Adicionalmente, técnicas de visualização baseadas em *rendering* volumétrico direto podem ser executadas em VTK.

Para a realização de *rendering* no VTK, objetos denominados atores (instâncias da classe vtkActor) representam e são associados aos diversos elementos que constituem uma cena. Cada um desses elementos é gerado pela transformação de dados de seu formato original para primitivas gráficas, seguindo o modelo *dataflow*. A sequência das transformações para a definição de um elemento, denominada *pipeline*, envolve objetos pertencentes a três tipos de classe: fontes (vtkSource), filtros (vtkFilter) e mapeadores (vtkMapper). As classes derivadas de vtkSource têm o objetivo de ler dados de fontes externas e gerar dados a partir de parâmetros estabelecidos. Exemplos de tais objetos são a definição de uma esfera (vtkSphereSource) e a leitura de uma poligonal a partir de arquivo (vtkPolyReader). As várias etapas para a manipulação e transformação dos dados envolvem objetos instanciados de classes derivadas de vtkFilter;

¹ Tcl é uma linguagem de comandos interpretada, e tem o Tk como extensão para o sistema X Windows.

conversão de formato dos dados e extração de superfícies são executados por filtros. Finalizando o *pipeline* de visualização, os dados obtidos são mapeados para primitivas gráficas por objetos relacionados com as classes derivadas de vtkMapper [Sch 96,97][Min 97].

Pelo exposto no parágrafo anterior, filtros oferecem mecanismos para que novos algoritmos sejam inseridos no contexto do VTK. Os filtros podem ser distinguidos pelos tipos de dados que aceitam na entrada e que produzem na saída, sendo esse par entrada/saída utilizado na organização dos tipos de filtro.

A estrutura de organização dos dados juntamente com seus atributos compõem, no VTK, um *dataset*, no qual podem estar definidas uma ou mais células, cada uma representando uma peça básica de construção, tal como polígono, poli-linha, linha e outras mais. Existem quatro tipos de *dataset* implementados:

1. vtkStructuredPoints. É representado implicitamente pelas dimensões (nx, ny e nz), razão de aspecto e origem. Os pontos são ordenados seguindo primeiro a direção x, depois a y e então a z. Corresponde a uma malha de dados cartesiana [Min 97];
2. vtkStructuredGrid. São definidas as dimensões do conjunto, e as coordenadas dos pontos são escritas explicitamente. A ordenação dos pontos é implícita como no caso anterior. Corresponde a malhas regulares, porém não igualmente espaçadas [Min 97];
3. vtkPolyData. As coordenadas dos pontos, bem como o tipo e a conectividade das células planares permitidas (um polígono representando um contorno, por exemplo), devem ser definidas explicitamente. Corresponde à definição de estruturas poligonais [Min 97];
4. vtkUnstructuredGrid. É o tipo de dados mais geral de todos, pois engloba todas as variedades de célula disponíveis, cujas configurações, juntamente com a definição das coordenadas dos pontos, devem ser fornecidas explicitamente. Células 3D, como hexaedros e tetraedros são permitidas nesse tipo de dados [Min 97].

Dados escalares, como temperatura, e vetoriais, como velocidade, são exemplos de atributos que podem ser associados aos vértices das células em cada um dos tipos de dados descritos acima [Sch 96,97].

Conceitos apresentados aqui serão utilizados para explicações da implementação no restante deste capítulo. Na próxima Seção é descrita a estrutura do projeto de reconstrução do dente virtual.

5.2 Estrutura do Projeto do Dente Virtual

O processo de reconstrução de um dente assume que as superfícies de interesse podem ser representadas por contornos em fatias paralelas e igualmente espaçadas ao longo de toda a

extensão do dente na direção z, conforme ilustrada na figura 5.1.

Figura 5.1 Contornos em fatias paralelas e igualmente espaçadas.

A figura 5.2 esquematiza os principais elementos do processo geral de criação de um modelo tridimensional de um dente. Os trechos indicados por setas pontilhadas representam partes opcionais da execução, e as duas setas maiores indicam que o processo pode, também, iniciar-se nesses pontos.

Cada contorno presente em uma determinada fatia é representado por um polígono descrito pelo seu respectivo conjunto de vértices. Esses vértices podem ser obtidos a partir da imagem correspondente à fatia a qual pertence o contorno. Tal imagem é resultante de um processo de digitalização de uma fotografia ou de um *frame* de um filme de vídeo. Existe, ainda, a possibilidade de usar dispositivos de aquisição que gerem diretamente as imagens, dispensando, dessa forma, a etapa conversão para o formato digital. Contornos provenientes de edições puramente manuais também foram utilizados nos testes dos processos de reconstrução (vide capítulo 6).

Para apoiar a etapa de RECONSTRUÇÃO DO MODELO, foi desenvolvido um filtro, denominado `vtkSamplingFromContoursFilter`, que produz uma malha volumétrica (`vtkStructuredPoints`) na qual pontos na vizinhança dos contornos têm como atributo o valor da distância do ponto ao contorno mais próximo. Dessa malha volumétrica é posteriormente extraída a superfície de interesse pelo método *Marching Cubes*, que já faz parte da biblioteca do VTK. À exceção da digitalização, as demais etapas exibidas nas caixas retangulares da figura 5.2 foram realizadas com programas complementares desenvolvidos neste trabalho. Esses programas estão descritos na Seção final deste capítulo.

Figura 5.2 Processo geral de criação de um modelo 3D de dentes.

O programa complementar de extração manual de contornos, usado na etapa EXTRAÇÃO DE CONTORNOS / REFERENCIAIS, permite, além da definição dos polígonos, a determinação das coordenadas de referenciais, que são pontos de controle comuns a todas as imagens. Devido a

variados fatores (posição relativa entre o objeto e a câmera nas várias tomadas de fotos, por exemplo), pode ocorrer um desalinhamento (rotação, translação) entre fatias diferentes, levando a uma reconstrução defeituosa. A partir dos referenciais, um outro programa complementar, o de correção geométrica, tem condições de calcular os parâmetros relacionados ao desalinhamento e aplicar uma transformação inversa, corrigindo, assim, o posicionamento relativo entre os polígonos que representam os contornos. Nem sempre os referenciais estão disponíveis, fazendo com que a etapa de CORREÇÃO GEOMÉTRICA fosse colocada como passo opcional na figura 5.2.

Caso os dados já estejam disponíveis como uma imagem no formato digital, a digitalização não é necessária e o processo de reconstrução se inicia a partir da IMAGEM. Essa eliminação de etapas também ocorre se os dados já estiverem disponíveis no formato numérico, ou seja, como um conjunto de polígonos. Esse é o caso do conjunto de dados enviados pelos colaboradores deste projeto na Inglaterra, que permitiu que o processo de geração do modelo 3D fosse executado a partir do ponto CONTORNOS da figura 5.2, após uma conversão entre formatos.

O conjunto de contornos extraídos precisa ser convertido para um formato válido do VTK. Uma escolha natural para o caso em questão, dentre os quatro formatos disponíveis apresentados na Seção 5.1, é o vtkPolyData ou *POLYGONAL DATASET*, que permite descrever os polígonos associados aos contornos. A conversão para o formato escolhido é feita por um programa complementar, na etapa de GERAÇÃO DE DADOS NO FORMATO VTK.

A interação com o dente virtual gerado pode ser feita no ambiente do próprio VTK, bem como por um *browser* capaz de manipular um arquivo VRML. Um arquivo VRML pode ser criado através de um objeto do VTK pertencente à classe vtkVRMLExporter.

A principal etapa desse processo é a reconstrução. Para tal, uma abordagem foi selecionada, adaptada e codificada no ambiente VTK. Sua implementação é descrita na próxima Seção.

5.3 Elaboração do Algoritmo

A abordagem de reconstrução escolhida para a geração de um modelo tridimensional para dentes, dentre aquelas apresentadas no capítulo 3, foi a de construção de superfícies utilizando funções de campo proposta por Jones e Chen [Jon 94], descrita na Seção 3.3. Essa é uma abordagem baseada em amostragem de distâncias dentro de um volume, na qual uma malha regular 2D é ajustada sobre cada fatia do volume. Em cada ponto dessa malha é armazenado um valor de distância entre o ponto da malha e o contorno mais próximo. As consecutivas malhas 2D são então “empilhadas” para formar uma malha regular 3D.

Além da disponibilidade de referências adicionais, a opção por essa abordagem baseou-se nas características, a ela associadas, listadas a seguir:

1. É requerido que os contornos, com relação às propriedades geométricas, sejam apenas fechados e simples;
2. Não é necessária a etapa de estabelecimento de correspondência entre contornos, presente em outras abordagens e que pode causar inconsistência e ineficiência;
3. É permitida a existência de múltiplos contornos em uma única fatia e a ocorrência de buracos.

Adicionalmente, foi considerada a aplicabilidade da abordagem ao problema de reconstrução de dentes, indicada pelo resultado obtido em um teste realizado utilizando o objeto `vtkImplicitModeller` do VTK que utiliza uma abordagem similar. Esse teste foi feito usando, como entrada, um conjunto gerado pela repetição de um contorno em quatro fatias consecutivas. Tal contorno foi formado pela extração de bordas a partir de uma foto dentária tirada ao microscópio e digitalizada. A imagem, o contorno e a superfície reconstruída estão apresentadas na figura 5.3.

Figura 5.3 Elementos do teste com `vtkImplicitModeller`.

(a) Imagem digitalizada a partir de foto ao microscópio.

(b) Contorno extraído da imagem.

(c) Superfície reconstruída. Detalhe da triangulação, no canto inferior direito, da área tracejada.

Nesse filtro é calculada a distância de cada ponto de um volume em relação a uma geometria de entrada (conjunto de contornos, em nosso caso); a menor distância obtida para cada ponto fica associada a ele. Com a saída desse filtro é possível gerar superfícies que estejam deslocadas em

relação à geometria de entrada.

Na abordagem escolhida é aplicado, também, o cálculo de distâncias e a escolha da menor entre elas; entretanto, valendo-se do fato de que os contornos são planares e as fatias paralelas, esse processamento é feito para cada fatia individualmente, ou seja, os pontos em uma fatia têm as distâncias estabelecidas em relação somente aos contornos na fatia onde eles estão situados. Consequentemente, obtém-se um melhor desempenho pela diminuição da quantidade de computação necessária.

Na subseção a seguir, são apresentados os principais detalhes da implementação da abordagem escolhida.

5.3.1 Aspectos e Elementos da Implementação do Filtro

A abordagem proposta é composta, de acordo com o exposto na Seção 3.3, por dois passos básicos: construção de volume e construção de superfície. No primeiro, é gerada uma representação volumétrica dos dados aplicando-se funções de campo sobre um conjunto de fatias que descrevem um objeto. No segundo passo, do volume composto é extraída a superfície de interesse.

A criação de um filtro implementando o primeiro passo (construção de um volume) foi suficiente para os nossos propósitos, pois o VTK tem disponíveis, em seu conjunto, algoritmos para o passo de extração de superfícies. Dentre eles, utilizamos o *Marching Cubes* para completar a reconstrução 3D.

O filtro implementado é composto por três partes principais:

1. Determinação do Posicionamento Relativo. É feita a verificação do posicionamento dos pontos em uma grade bidimensional em relação aos contornos presentes nessa grade/fatia (cada ponto é classificado como externo a todos os contornos, sobre ou interno a um contorno);
2. Cálculo da Menor Distância. Para cada ponto de uma fatia (grade) é calculada sua distância à aresta mais próxima, ou seja, a distância ao contorno mais próximo;
3. Fechamento de contornos. Pode ser realizado o fechamento de contornos na primeira e na última fatia, caso seja necessário.

Os itens 1 e 2 correspondem ao primeiro passo da abordagem selecionada, e o item 3 é uma extensão. Cada uma dessas partes está explicada nas subseções a seguir.

5.3.1.1 Determinação do Posicionamento Relativo

A primeira parte do filtro, que é determinar a posição dos pontos da grade sobre uma fatia em relação aos respectivos contornos, foi implementada utilizando-se um algoritmo de preenchimento de polígonos que aplica coerência de arestas com linhas de varredura e usa estruturas de lista de arestas e lista de arestas ativas [Fol 96] [Tra 95]. É importante observar que, no referido algoritmo, alguns pontos do polígono não são traçados, como por exemplo, pontos mínimos locais e pontos sobre algumas arestas horizontais. Tais pontos foram considerados no algoritmo implementado, pois todos os pontos dos contornos devem estar definidos, isto é, os pontos não considerados pelo algoritmo de preenchimento são armazenados e processados à parte.

O processamento de uma determinada fatia envolve, primeiramente, a determinação dos contornos pertencentes à ela (mesmo valor de coordenada z) e a criação de uma correspondente **lista de arestas**. Essa lista deve estar organizada em ordem decrescente de linha de varredura (mesma ordem de percorrimento das linhas de varredura) e, da esquerda para a direita com respeito a posição relativa das arestas no plano da fatia (arestas posicionadas mais a esquerda são processadas primeiro). Essa lista alimenta a **lista de arestas ativas**, que contem aquelas arestas que interceptam a linha de varredura que está sendo manipulada. À medida que a linha de varredura caminha, as arestas que passam a interceptá-la são transferidas da lista de arestas para a de arestas ativas e as que deixam de sofrer interseção são retiradas da lista de arestas ativas. Os elementos de ambas as listas, isto é, arestas e arestas ativas, têm a mesma estrutura, descrita na figura 5.4.

y _{max}	Primeira linha de varredura que intercepta esta aresta.
dy	Número de linhas de varredura que esta aresta intercepta. Decrementado a cada iteração.
x _{trace}	Coordenada x do ponto de interseção com a linha de varredura atual.
den (Δy) num dx	Utilizados no cálculo do novo ponto de interseção (x_{trace}) empregando somente aritmética de inteiros. Enquanto $num > den$, x_{trace} deve ser incrementado. O valor de num sofre variações de dx em dx .
x _{min}	Junto com a coordenada da última linha de varredura desta aresta, define o ponto mínimo. Se for um ponto mínimo local, será traçado posteriormente.

Figura 5.4 Estrutura de um elemento da lista de arestas e da lista de arestas ativas.

Na figura 5.5 é apresentada uma ilustração de como a lista de arestas e a lista de arestas ativas se relacionam.

Figura 5.5 Relacionamento entre lista de arestas e lista de aresta ativas.

A coerência de aresta é empregada no cálculo da interseção de uma aresta com a linha de varredura seguinte, levando em consideração a interceptação com a linha atual. Pela equação reduzida da reta temos que

$$y_i = x_i * \frac{\Delta y}{\Delta x} + b \quad \Rightarrow \quad x_i = \frac{\Delta x}{\Delta y} * (y_i - b), \quad \Delta x \text{ e } \Delta y \neq 0.$$

a variável y , que corresponde a linha de varredura, sofrer um incremento unitário, então obtemos

$$x_{i+1} = \frac{\Delta x}{\Delta y} * (y_i + 1 - b), \text{ ou seja, } \boxed{x_{i+1} = x_i + \frac{\Delta x}{\Delta y}}.$$

Dessa forma, a coordenada x do ponto de interseção de uma aresta com a próxima linha de varredura pode ser obtida através de um incremento ($\Delta x / \Delta y$, constante) à coordenada atual.

Utilizando-se esse algoritmo de preenchimento de polígonos, a determinação da posição relativa dos pontos da grade é obtida pelas seguintes regras:

1. Os pontos de interseção das arestas com as linhas de varredura (x_{trace}) são pontos sobre os contornos;

2. Os pontos entre duplas de pontos de contornos, tomados dois a dois da esquerda para direita, estão na parte interna de um contorno;
3. Os demais pontos são externos.

A figura 5.6 ilustra o relacionamento entre pontos e contornos.

Figura 5.6 Relacionamento entre pontos e contornos.

Os pontos não considerados pelo algoritmo são classificados como pertencentes ao contorno (pontos mínimos locais e pontos sobre algumas arestas horizontais).

Após a determinação da posição relativa dos pontos é executado o cálculo de distância, que é descrito a seguir.

5.3.1.2 Cálculo da Menor Distância

Para o cálculo de distância, que corresponde à segunda parte da codificação do filtro, utiliza-se a distância euclidiana. Se a projeção ortogonal do ponto está sobre a aresta (segmento de reta), então é empregada a fórmula de distância entre ponto e reta. Caso contrário, calcula-se a distância do ponto a extremidade da aresta que está mais próxima (ver figura 5.7). O cálculo é realizado em relação a todas as arestas e a menor distância é escolhida como valor associado ao ponto em questão.

Figura 5.7 Esquema do cálculo de distância na segunda parte do filtro.

As fórmulas matemáticas empregadas no cálculo das distâncias e na determinação das coordenadas da projeção ortogonal [Lap 83] são:

$$d_{\text{ponto-ponto}} = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$$

e

$$d_{\text{ponto-reta}} = \frac{|a * x_P + b * y_P + c|}{\sqrt{a^2 + b^2}}$$

e

$$y = \frac{b * c - a^2 * y_P + a * b * x_P}{a^2 + b^2},$$

$$x = -\frac{b * y + c}{a}.$$

onde (x_A, y_A) , (x_B, y_B) e (x_P, y_P) são as coordenadas dos pontos extremos da aresta e do ponto cuja distância será calculada, respectivamente.

Considerando-se que, na sequência, é empregado o algoritmo *Marching Cubes* para extração da superfície, e de acordo com argumentos apresentados em [Jon 94], é possível reduzir o número de pontos que necessitam ter a distância calculada àqueles que são vértices dos sub-volumes nos quais alguma parte da superfície está contida. Assim, tomando-se duas fatias adjacentes, busca-se os pontos que são vértices de cubos pelos quais a superfície passa (ver figura 5.8). A busca desses pontos pode ser reduzida àquela parte do volume total que engloba todos os contornos, ou seja, a região próxima às bordas de cada fatia pode ser ignorada. A condição do cubo conter ou não a superfície é dada pela regra: se os vértices de uma célula estão todos fora (valor -1) ou todos dentro (valor 1) da superfície, então a superfície não intercepta a célula, caso contrário, o sub-volume é interceptado pela superfície.

Figura 5.8 Vértices de um cubo formado por duas fatias adjacentes.

Além dos dois passos descritos até aqui, no filtro implementado é considerada a necessidade de fechamento de contornos presentes na primeira e na última fatia. Esse fechamento fica a critério do usuário do filtro. A estratégia adotada é apresentada na próxima subseção.

5.3.1.3 Fechamento de Contornos

Casos nos quais seja necessário o fechamento dos contornos da primeira e/ou última fatia podem ocorrer com frequência. No filtro implementado, a estratégia adotada foi a criação de uma fatia adicional, precedendo a primeira ou seguindo a última fatia do conjunto. Os valores de distância para os vértices dessas novas fatias são estipulados de tal forma que provoque o fechamento do contorno, quando do traçado da superfície pelo método *Marching Cubes*. Esses valores devem ser negativos, em contraposição aos positivos dos pontos internos do contorno que se pretende fechar. A variação da magnitude desses valores permite controlar a localização, dentro do espaço entre a fatia original e a adicional, da superfície de fechamento. A partir de um valor p , no intervalo $(0,1)$, indicativo da posição do ponto mais alto da superfície de fechamento, um único valor de distância para todos os pontos da fatia adicional é calculado pela fórmula:

$$d_{\text{adicional}} = -\frac{(1-p)}{p} * d_{\text{original_max}}.$$

Vale notar que quanto menor o valor de p , mais próxima ao contorno estará a superfície de fechamento. Observe a figura 5.9.

Figura 5.9 Definição da posição da superfície de fechamento.

Para o desenvolvimento da fórmula anterior, seja d_a o valor relacionado com a fatia adicional e d o valor de um ponto da fatia original. Observando a figura 5.9 e lembrando que $d > 0$ e $d_a < 0$, temos que

$$d - p * (d - d_a) = 0 \Rightarrow d_a = -\frac{(1-p)}{p} * d.$$

Se d assumir o valor da maior distância calculada na fatia original, então p indica a localização do ponto mais alto da superfície de fechamento.

Em seguida, é apresentada a lógica e os detalhes do filtro implementado, cujos pontos principais foram descritos até aqui.

5.4 Lógica e Detalhes do Filtro Implementado

As figuras 5.10, 5.11 e 5.12 apresentam a lógica de cada uma das partes do filtro implementado, descritas na Seção 5.3.

```
<<< Fase I : Determinação da posição dos pontos da grade em relação aos contornos >>>

Marcar todos os pontos de grade do volume como externos (valor -1);
Enquanto (houver contornos a processar)
  Verificar quais são os contornos da fatia em análise;
  Montar a Lista de Arestas - LA ; (ordenada por linha de varredura e posição relativa na fatia)
    (marcar as arestas horizontais)
  Inicializar a Lista de Arestas Ativas - LAA ;(transferir as arestas da linha de varredura max)
  Para (cada linha de varredura)
    Marcar os pontos do contorno e os internos de acordo com a LAA;
    Calcular as interseções das arestas em LAA com a próxima linha de varredura;
      (marcar os pontos mínimos locais)
      (retirar as arestas que deixam de sofrer interseção)
    Verificar se é necessário transferir arestas de LA para LAA;
      (manter a ordem de posição relativa entre arestas - esquerda para direita)
  Fim-Para
  Classificar os pontos sobre arestas horizontais e os pontos mínimos locais;
    (pertencentes ao contorno)
Fim-Enquanto
```

Figura 5.10 Lógica da fase de determinação da posição relativa dos pontos da grade.

```
<<< Fase II: Calculo das distâncias >>>

Montar uma lista das retas referentes a primeira fatia;
Para (fatias da segunda a penúltima)
  Montar uma lista das retas da fatia adjacente a atual;
  Para (células formadas pelas duas fatias)
    Se (superfície intercepta a célula) então
      Calcular o valor de distância para os vértices da célula;
    Fim-Se
  Fim-Para
  Marcar a fatia adjacente como a atual;
Fim-Para
```

Figura 5.11 Lógica da fase de cálculo de distâncias por onde a superfície passa.

```
<<< Preparação para o fechamento da superfície , se necessário >>>

Se (fechar superfície) então
  Criar uma nova fatia;
  Calcular e atribuir o valor de distância para os pontos da fatia adicional;
Fim-Se
```

Figura 5.12 Lógica para o fechamento da superfície.

O filtro VTK implementado, `vtkSamplingFromContoursFilter`, deve aceitar um `vtkPolyData` (polígonos/contornos) como entrada e produzir um `vtkStructuredPoints` (malha regular 3D) na saída. Ele foi derivado da classe `vtkDataSetToStructuredPointsFilter`, pois essa classe tem como parâmetro de saída um `vtkStructuredPoints`, gerado a partir de um `vtkDataset`, do qual `vtkPolyData` é derivado. O apêndice B apresenta as observações ao formato `vtkPolyData` necessárias à execução do filtro `vtkSamplingFromContoursFilter`.

Os métodos implementados no filtro `vtkSamplingFromContoursFilter` permitem as seguintes operações:

1. Estabelecer a resolução de amostragem nas direções x e y;
2. Definir a razão de aspecto do volume de saída;
3. Optar pelo fechamento de contornos.

Esses itens estão descritos em detalhes no apêndice A, que apresenta uma página HTML de manual para o `vtkSamplingFromContoursFilter`, nos moldes do manual do VTK.

Para acesso ao filtro, que foi concebido para manipular coordenadas com valores inteiros positivos (caso natural de imagens exibidas no monitor), foi criada uma biblioteca de ligação dinâmica (*Dynamic Link Library* - DLL), que deve ser usada em conjunto com a dll do VTK (`vtkdll.dll`).

A figura 5.13 mostra um exemplo de uso do filtro, com alguns trechos de um programa em C++. Em negrito está destacado o trecho referente ao `vtkSamplingFromContoursFilter` com a configuração de seus vários parâmetros e a verificação do fechamento ou não de contornos. Os dados de entrada vêm de um objeto, o `vtkPolyReader`, que lê um *polygonal dataset*. Na sequência do filtro é feita a extração da superfície com o `vtkMarchingCubes`, que é posteriormente mapeada para primitivas gráficas pelo `vtkPolyMapper`. Após a execução do *pipeline* de visualização, a superfície é associada a um ator (`vtkActor`) que será apresentado numa cena durante o processo de *rendering*.

Para a extração da superfície pode ser utilizada uma outra técnica que utilize um critério local similar ao do *Marching Cubes*, ou seja, a mesma forma de verificar a interseção entre célula e superfície.

Nesta Seção foram apresentados detalhes da implementação do filtro `vtkSamplingFromContoursFilter`. Vários programas auxiliares foram desenvolvidos para apoiar a aplicação do programa com o filtro de reconstrução implementado para o caso da reconstrução dentária. Esses programas são descritos na próxima Seção.

```

#include <vtk.h>
#include "c:\mestrado\c\supd\vtkSamplingFromContoursFilter.h"

// leitura do Polygonal Dataset ou vtkPolyData
vtkPolyReader *SlicesFile = new vtkPolyReader;
SlicesFile->SetFilename(FileId);
SlicesFile->Update();

// aplicação do filtro para gerar um volume com as distâncias aos contornos
vtkSamplingFromContoursFilter *DistanceGrid = new vtkSamplingFromContoursFilter;
DistanceGrid->SetInput(SlicesFile->GetOutput()); // obtenção do conjunto de polígonos lidos por SlicesFile
DistanceGrid->SetSliceSamplingResolution(10,10); // resolução de amostragem: 10 pontos em x e y (z é sempre 1)
DistanceGrid->SetOutputAspectRatio(1,1,3); // razão de aspecto do volume de saída, nas direções x, y e z
// essa informação não é extraída dos dados de entrada
DistanceGrid->SetOutputOrigin(0,0,0); // origem do volume de saída
// permite o correto posicionamento das várias superfícies, se houver

if (close11) {
    DistanceGrid->CloseFirstSurfaceOn(); // habilita o fechamento de contornos na primeira fatia
    DistanceGrid->SetCloseFirstSurfaceZero(0.01); // posição do ponto mais alto da superfície de fechamento
}

if (close12) {
    DistanceGrid->CloseFirstSurfaceOn(); // habilita o fechamento de contornos na última fatia
    DistanceGrid->SetCloseFirstSurfaceZero(0.01); // posição do ponto mais alto da superfície de fechamento
}

// criação da superfície
vtkMarchingCubes *Contours = new vtkMarchingCubes;
Contours->SetInput(DistanceGrid->GetOutput());
Contours->SetValue(0,0,0); // a superfície de interesse possui escalar (distância) igual a 0

// mapeamento da superfície para primitivas gráficas
vtkPolyMapper *SurfaceMapper = new vtkPolyMapper;
SurfaceMapper->SetInput(Contours->GetOutput());
SurfaceMapper->ScalarsVisibleOff();

// associação a um ator para inclusão na cena e posterior rendering
vtkActor *SupActor = new vtkActor;
SupActor->SetMapper(SurfaceMapper);
SupActor->GetProperty()->SetColor(1,1,1);
SupActor->GetProperty()->SetOpacity(0.5);

```

Figura 5.13 Exemplo de uso do filtro *vtkSamplingFromContoursFilter* em C++.

5.5 Funcionalidade de Programas Complementares

Juntamente com o programa de reconstrução utilizando o filtro implementado, outros programas complementares foram codificados, os quais tem as seguintes funções:

1. Extração manual de contornos e determinação de referenciais;
2. Correção geométrica de um conjunto de contornos extraídos;
3. Geração de arquivo no formato do VTK a partir de um conjunto de contornos;

4. Interface para estabelecer os parâmetros da reconstrução de um dente.

Esses programas estão descritos a seguir através de sua interface gráfica e de uma explicação resumida do funcionamento. O relacionamento entre tais programas foi apresentada na figura 5.2.

5.5.1 Extração Manual de Contornos e Referenciais

Figura 5.14 Interface gráfica do programa de extração manual de contornos e referenciais.

A figura 5.14 ilustra uma janela do programa de edição de contornos. Sobre uma imagem de um objeto, previamente carregada, é permitido marcar as coordenadas dos vértices dos contornos (bordas) que definem alguma parte de interesse. Essas coordenadas ficam vinculadas a uma determinada fatia pelo fornecimento de um valor para a coordenada z (contornos numa mesma fatia têm o valor de z iguais). Para auxiliar no traçado de um contorno está disponível uma janela de zoom numa vizinhança do cursor (janela menor na figura 5.14), sendo o tamanho dessa

vizinhança controlado por um fator de zoom dado pelo usuário. Os contornos podem ser gravados em um arquivo no formato texto (ASCII), e também no formato bitmap do Windows (bmp). Existindo pontos de referência na imagem, suas coordenadas podem ser marcadas e salvas no disco em formato ASCII, conforme instruções fornecidas em tela. Esse programa foi utilizado para editar a maioria dos contornos dos casos de teste do capítulo 6.

5.5.2 Correção Geométrica

Figura 5.15 Interface gráfica do programa de correção geométrica.

A figura 5.15 apresenta a interface do programa de correção geométrica, que toma um arquivo de referenciais e um conjunto de observações e gera um arquivo de parâmetros para o alinhamento dos contornos. A correção implementada é a transformação isogonal, descrita no capítulo 2. A partir de um conjunto de pontos (chamados referenciais), tomados como corretamente posicionados, além de um conjunto de pontos observados (observações) relacionados aos referenciais, são calculados os parâmetros que descrevem o desalinhamento entre os dois conjuntos. Tais parâmetros são a translação em x, a translação em y, a rotação e a escala. De posse dos parâmetros é possível definir a transformação inversa que posiciona corretamente pontos observados. No problema de reconstrução em questão, pontos na primeira fatia são tomados como referenciais e os pontos correspondentes nas demais fatias formam os vários

conjuntos de pontos observados, cada conjunto proveniente de uma dada fatia. Os referenciais marcados durante a execução do programa de extração (subseção 5.5.1) permitem calcular os parâmetros que possibilitam a correção das coordenadas dos vértices dos contornos. O efeito da aplicação da correção geométrica será discutido no capítulo 6.

5.5.3 Geração de Arquivo no Formato VTK

Figura 5,16 Interface gráfica do programa de geração de arquivo no formato VTK.

Contornos extraídos e (opcionalmente) corrigidos devem ser agrupados e transformados num formato que possa ser lido pelo VTK. Unificando os vários conjuntos de contornos, esse programa gera um arquivo com um formato válido no VTK, tipo POLYGONAL DATASET (*vtkPolyData*). Um exemplo de arquivo é apresentado no apêndice B.

5.5.4 Interface para Realizar a Reconstrução

Na figura 5.17 são apresentados os parâmetros da interface para o programa de reconstrução. Esses parâmetros controlam as características de cada elemento (superfície) que compõe o objeto reconstruído. As características se referem aos seguintes itens: arquivo de dados que descreve o elemento, resolução de amostragem, razão de aspecto, posicionamento, altura da superfície de fechamento (se existir), cor e opacidade da superfície. A lista de elementos de visualização pode ser gravada para disco, editada e lida do disco. A ordem de inclusão de uma superfície na cena tem importância, especialmente quando algum grau de transparência está envolvido, na medida em que o VTK utiliza essa ordem para realizar o *rendering*. A superfície mais externa deve ser “renderizada” por último, de forma a permitir que superfícies em seu interior permaneçam visíveis. Um arquivo no formato VRML pode ser gerado a escolha do usuário. A visualização da cena definida por esse conjunto de parâmetros pode ser testada interativamente.

Figura 5.17 Interface gráfica do programa de interface para reconstrução.

Com a elaboração e implementação do filtro `vtkSamplingFromContoursFilter`, no contexto do VTK, somada à codificação dos programas complementares, foram realizados testes de desempenho e de resultados visuais. O próximo capítulo descreve esses testes, mostra e analisa os resultados, e apresenta as conclusões deste trabalho.

6. RESULTADOS E CONCLUSÕES

Com a elaboração e implementação do algoritmo selecionado como um filtro (`vtkSamplingFromContoursFilter`) nos moldes do VTK e a codificação dos diversos programas complementares, vários testes foram realizados para a avaliação das características do algoritmo.

Os resultados obtidos nos testes foram avaliados, principalmente, sob dois aspectos:

1. Desempenho: tempo de processamento da reconstrução, sem considerar o tempo de *rendering*;
2. Visual: resultado final decorrente do *rendering* da superfície reconstruída.

Os testes foram organizados de acordo com duas abordagens:

1. Interna: avaliação dos resultados obtidos pela variação do parâmetro de resolução de amostragem no `vtkSamplingFromContoursFilter`;
2. Externa: comparação dos resultados obtidos pela aplicação do `vtkSamplingFromContoursFilter` com aqueles obtidos pelo uso do `vtkImplicitModeller`.

Neste capítulo são apresentados as características e os objetivos dos testes realizados, a avaliação dos resultados obtidos e as conclusões decorrentes. Sugestões de trabalhos futuros também são apresentadas.

6.1 Testes Realizados

Os testes foram executados com três conjuntos de dados, contendo uma ou mais superfícies descrevendo as várias partes de um dente. A seguir, na tabela 6.1, são apresentadas uma descrição

e as características de cada conjunto.

Conjunto	Descrição	Nº de Superfícies	Nº de Pontos	Nº de Polígonos	Nº de Fatias
1. Desenho Manual	Digitalização de desenho à mão, numa folha de papel, com diversos contornos	3	278 376 35	11 9 3	11 9 3
2. Dente em Molde	Digitalização de um fita de vídeo contendo tomadas correspondentes às faces desgastadas, com uma lixa de disco, de um dente real incluso num molde cúbico de acrílico	2	923 1604	15 25	15 15
3. Dados editados, fornecidos pelos Colaboradores da Inglaterra	Conjunto de dados gerados pela conversão de dados enviados pelos colaboradores da University of East Anglia	1	1257	61	22

Tabela 6.1 Descrição dos conjuntos de dados utilizados nos testes.

Os polígonos correspondentes aos contornos dos conjuntos 1 e 2 foram extraídos no programa complementar de extração de contornos. Os dados originais do conjunto 3 foram convertidos para valores inteiros positivos para adequação ao uso com o filtro `vtkSamplingFromContoursFilter`. Usando um outro programa complementar, o de geração de arquivo no formato VTK, a partir desses três conjuntos foram gerados os arquivos (tipo `vtkPolyData`) de entrada para os testes.

O conjunto de dados 1 foi utilizado para testar a aplicabilidade do `vtkSamplingFromContoursFilter` na reconstrução de detalhes de um dente; com o conjunto 2 foram feitos testes internos do filtro implementado, e foram avaliados os resultados da aplicação do processo correção geométrica das possíveis distorções nos dados; e, finalmente, com o conjunto 3 realizou-se uma comparação da abordagem implementada com o `vtkImplicitModeller` do próprio VTK. Além dos quesitos visual e de desempenho, foram analisados a quantidade de polígonos da superfície reconstruída e o arquivo VRML produzido.

Os programas usados nos testes foram codificados na linguagem C++ e executados num microcomputador com a seguinte configuração:

- processador Pentium de 166 MHz;
- memória cache L2 de 256 KB;
- memória RAM de 32 MB;
- Sistema Operacional Windows 95.

Além disso, o filtro `vtkSamplingFromContoursFilter` ficou definido numa biblioteca de ligação dinâmica (DLL, com uma biblioteca de referências externas, `.lib`, correspondente) separada da biblioteca do VTK (`vtkdll.dll` e `vtkdll.lib`).

Na medida de desempenho, o tempo foi computado no processamento da seguinte sequência:

`vtkPolyReader` → {`vtkSamplingFromContoursFilter` | `vtkImplicitModeller`} →
→ `vtkMarchingCubes` → `vtkPolyMapper` → `vtkActor`.

Ou seja, o tempo considerado foi o gasto para realizar a leitura dos dados, gerar o volume com as distâncias, extrair a superfície, fazer o mapeamento para primitivas gráficas e realizar a associação com um ator. Nos testes foram feitas três medidas de tempo e calculada a média.

Nas subseções seguintes são apresentados os testes e seus respectivos resultados. As análises e observações de caráter mais geral estão apresentadas na Seção 6.2.

6.1.1 Reconstrução de Detalhes do Dente

A figura 6.1 mostra a imagem obtida pela digitalização de um desenho à mão contendo vários contornos correspondentes a duas partes de um dente, coroa e raiz.

Figura 6.1 Desenho dos contornos, à mão, digitalizado.

O objetivo desse teste foi verificar se os detalhes (câmara pulpar e relevo da coroa, por exemplo) de um dente seriam bem reconstruídos com o uso do filtro `vtkSamplingFromContoursFilter`. Além

das duas partes presentes no desenho, mais alguns contornos, relativos a um nervo, foram criados no processo de extração daqueles contornos, fazendo o conjunto 1 ser composto por três superfícies.

Na figura 6.2 estão apresentados os resultados visuais da reconstrução com a utilização dos dados do conjunto 1.

Figura 6.2 Reconstrução com os dados do conjunto 1.
Visão completa com as três superfícies: coroa, raiz e nervo (direita).
Destaque da entrada do canal na base da raiz (esquerda).

Comparando a figura 6.1 e a figura 6.2(direita) observamos que o formato, as partes e os detalhes do hipotético dente foram bem reproduzidos. A seta na figura aponta para a câmara pulpar (canal), que pode ser percebida devido a transparência dada a raiz. Essa câmara, delimitada pela parede interna da raiz, é onde se alojam os nervos e vasos sanguíneos, e possui uma entrada na base da raiz (destacado por elipses nas figuras). O formato ou relevo da coroa, que é a parte visível do dente na boca, também estão mostrados na figura 6.2(direita). A figura 6.3 também ilustra o relevo da coroa.

O filtro implementado permite, também, que a superfície formada em cada conjunto de contornos seja fechada ou não, em cada um dos seus extremos. A figura 6.3 mostra a coroa do conjunto 1 reconstruída com e sem fechamento. A seta na figura 6.3(b) aponta a parte que ficou aberta.

Figura 6.3 Reconstrução com e sem fechamento da superfície.
(a) Coroa reconstruída com fechamento.
(b) Coroa reconstruída sem fechamento.

Os resultados obtidos com esse teste demonstraram que o filtro implementado se aplica bem ao problema de geração de um modelo 3D para dentes a partir de contornos planares. A subseção seguinte descreve os testes que avaliaram os resultados da aplicação do filtro utilizando como entrada os dados do conjunto 2.

6.1.2 Análise Interna e Aplicação de Correção Geométrica

O conjunto 2 é composto por dados relativos a duas superfícies: o esmalte (presente na coroa) e uma parte da dentina (contida na raiz). Os polígonos, correspondentes aos contornos, foram extraídos a partir de imagens obtidas pela digitalização de quadros de uma fita de vídeo. Um dente real foi inserido num molde acrílico de formato cúbico, juntamente com quatro hastes de grafite para controle do processo de correção geométrica. O molde foi desgastado com uma lixa de disco acoplada a um micromotor utilizado por dentistas. À medida que novas fatias do dente apareciam, elas eram filmadas. Na figura 6.4 são apresentados um dente no molde com as marcas das hastes de grafite e os correspondentes contornos extraídos.

Figura 6.4 Dente em molde com hastes de grafite e contornos correspondentes.
(a) Face de um dente desgastado. Círculos mostram os pontos (grafites) de referência.
(b) Contornos extraídos a partir da imagem à esquerda.

Os dados do conjunto 2 permitiram que fossem avaliados: o tempo de processamento, o produto visual com a variação da resolução de amostragem, e as implicações da aplicação do processo de correção geométrica aos contornos. A correção foi possível devido à colocação de pontos de controle (destacados por círculos na figura 6.4(a)), presentes em todas as imagens.

A tabela 6.2 apresenta os valores obtidos com relação ao tempo gasto no processamento, considerado desde a leitura dos dados até a associação com o ator, de acordo com a resolução de amostragem. A coluna 1 corresponde às resoluções de amostragem empregadas nos testes; nas colunas 2 e 3 estão os valores da quantidade de pontos do volume de saída e do número de

polígonos gerados, respectivamente, discriminados por superfície e totalizados; na coluna 4 são mostrados os tempos médios calculados pela média aritmética de 3 medidas.

(1) Resolução	(2) Dimensão do Volume de Saída Superfície 1 Superfície 2 Total	(3) Nº de Polígonos Gerados Superfície 1 Superfície 2 Total	(4) Tempo Médio (seg)
2 x 2	187 x 186 x 15 = 521730 182 x 175 x 15 = 477750 999480	19005 35404 54409	16,33
3 x 3	125 x 124 x 15 = 232500 122 x 117 x 15 = 214110 446610	9850 18265 28115	8,97
5 x 5	76 x 75 x 15 = 85500 74 x 71 x 15 = 78810 164310	4485 8294 12779	4,78

Tabela 6.2 Resultados obtidos pela variação da resolução de amostragem.

A partir dos números na tabela 6.2, foi elaborada a tabela 6.3, observando o relacionamento de tais números. A coluna 1 descreve quais são as linhas da tabela 6.2 consideradas; os valores das colunas 2 e 3 são resultantes da divisão do total de uma linha pelo total da outra que está sendo comparada; a coluna 4 é a relação entre os tempos médios; a coluna 5 é o resultado da divisão da coluna 3 pela 2.

(1) Relação	(2) Dimensão do Volume de Saída	(3) Nº de Polígonos Gerados	(4) Tempo Médio	(5) Polígonos / Volume
(2x2) / (3x3)	2,24	1,94	1,82	0,05
(2x2) / (5x5)	6,08	4,26	3,42	0,06
(3x3) / (5x5)	2,72	2,20	1,88	0,08

Tabela 6.3 Relação entre os valores da tabela 6.2.

Observando-se as colunas 2, 3 e 4, percebe-se que uma aumento na resolução de amostragem provoca um tempo de processamento proporcionalmente menor em relação ao volume de saída e ao número de polígonos gerados. Por exemplo, na primeira linha (comparação entre as resoluções 2x2 e 3x3) o volume de saída é 2,24 vezes maior, enquanto o número de polígonos gerados é 1,94 vezes maior, e o tempo é ainda menor, 1,82 vezes. O mesmo raciocínio se aplica às outras duas linhas. Na coluna 5, nota-se que a relação entre o número de polígonos e o volume de saída é também inversamente proporcional ao aumento da resolução. Esse comportamento era previsível, uma vez que, na reconstrução, são considerados pontos que se situam na vizinhança dos contornos, ou seja, muitos dos pontos adicionados pelo aumento da resolução de amostragem

não exercem papel ativo no processo de reconstrução pela abordagem que estamos testando¹.

Com relação ao aspecto visual, os resultados obtidos estão mostrados na figura 6.5.

Figura 6.5 Resultados da reconstrução usando dados do conjunto 2.

(a) Resolução de amostragem 5x5.

(b) Resolução de amostragem 3x3.

(c) Resolução de amostragem 2x2.

(d) Reconstrução após correção geométrica. Resolução 2x2.

Pela inspeção da área delimitada pelas áreas retangulares das imagens a, b e c da figura 6.5, percebe-se que, a medida em que há um aumento na resolução de amostragem, o dente resultante ganha um aspecto mais blocado, menos suave. Esse resultado pode estar relacionado com o formato que os contornos são apresentados, especialmente nos trechos curvos, uma vez que uma resolução de amostragem mais fina implica numa definição mais precisa do contorno, ficando mais próxima ao original. Esse ponto, além de outros referentes à figura 6.5 serão apreciados na Seção 6.2.

O conjunto 2 foi obtido a partir de uma fita de vídeo. No procedimento de filmagem das várias faces de um dente à medida que ele era desgastado, o molde de acrílico era colocado numa determinada posição e filmado. Para cada fatia, o molde era retirado dessa posição, desgastado e reposicionado para uma nova filmagem. O reposicionamento, apesar de obedecer a marcações de posição, pode gerar um desalinhamento entre imagens, levando a reconstruções imprecisas.

Para mapear e corrigir esse desalinhamento foi implementada a transformação isogonal (ver capítulo 2) na forma de programa complementar. Nessa transformação, assume-se que ocorreram duas translações (direções x e y), além de uma rotação e uma escala. A partir daí são calculados, com base em alguns pontos de controle, os parâmetros que caracterizam essas distorções para aplicação de uma transformação inversa. No caso desse teste, cada imagem tinha quatro pontos

¹ Deve ser observado que nas medições mencionadas aqui não estão incluídos o tempo para a execução do *rendering*, e um maior número de polígonos acarreta um tempo de *rendering* maior.

de controle (chamados referenciais - ver círculos na figura 6.4(a)); a primeira imagem foi tomada como base e as demais foram ajustadas à ela. A tabela 6.4 mostra os parâmetros calculados pelo programa para as 15 imagens (fatias) do conjunto de dados 2.

	translação em x	translação em y	rotação (graus)	escala
1	0.000000	0.000000	0.000000	1.000000
2	-9.798332	0.747961	0.248855	1.009839
3	-2.837387	-6.722504	0.470051	1.011182
4	-2.839253	4.615165	0.140702	1.009873
5	-0.158042	4.783399	0.379906	1.004636
6	-6.475258	10.294084	0.456132	1.005364
7	-5.143038	6.264398	0.451694	1.013539
8	-7.063005	4.961224	0.342721	1.013213
9	-15.838334	6.028474	0.283221	1.009506
10	-2.593812	9.751938	0.350979	0.988827
11	-9.976556	3.339468	0.253297	1.015661
12	-6.258431	7.073794	0.420258	1.007457
13	-8.791644	4.625638	0.276223	1.007931
14	-19.810544	23.498701	1.702355	1.000284
15	-7.810715	10.388184	0.882450	0.995564

Tabela 6.4 Parâmetros da transformação Isogonal relativos ao conjunto de dados 2.

Como a primeira imagem é considerada base para as demais, a primeira linha mostra que não há nenhuma correção a ser feita. Pelas linhas restantes, nota-se que não houve grandes variações na escala, ficando seus valores muito próximos a 1, assim como foram pequenos os ângulos de rotação, podendo ter ocorrido pelo fato da superfície, na qual se apoiava o molde, não ser perfeitamente plana. Já os valores referentes à translação são consideráveis em função do reposicionamento a cada desgaste para tomada da imagem. Um outro fator que influencia esses parâmetros é a "pontaria" ou marcação dos referenciais, ou seja, uma possível imprecisão na determinação desses pontos pelo operador.

A figura 6.5(d) representa o resultado da reconstrução com os contornos corrigidos, seguindo os dados da tabela 6.4, e que foi realizada com os mesmos parâmetros utilizados na imagem da figura 6.5(c). Comparando as referidas figuras, observa-se que o canal, indicado pelas setas, ficou mais fino e proporcional ao outro canal; e a coroa, destacada pelas elipses, ficou mais próxima à raiz. Além disso, a silhueta do dente ficou mais alongada e a superfície reconstruída ficou mais suavizada. A tabela 6.5 mostra que, também, numericamente o resultado melhorou.

A subseção seguinte apresenta os aspectos do teste de comparação com outro filtro, o `vtkImplicitModeller`, disponível no VTK.

(1) Resolução 2 x 2	(2) Dimensão do Volume de Saída Superfície 1 Superfície 2 Total	(3) Nº de Polígonos Gerados Superfície 1 Superfície 2 Total	(4) Tempo Médio (seg)
Sem Correção	187 x 186 x 15 = 521730 182 x 175 x 15 = 477750 999480	19005 35404 54409	16,33
Com Correção	185 x 181 x 15 = 502275 183 x 172 x 15 = 472140 974415	16643 28806 45449	15,20

Tabela 6.5 Dados da reconstrução sem e com correção geométrica.

6.1.3 Comparação com o vtkImplicitModeller

A partir de dados fornecidos pelos colaboradores da *School of Information System* da *University of East Anglia* na Inglaterra, gerados a partir de um editor de contornos (ver [Goe 98]), foi feita uma adaptação desses conjuntos de dados para o formato aceito no `vtkSamplingFromContoursFilter` através da conversão de seus valores para inteiros positivos e deslocamento para a origem, gerando o conjunto de dados 3 usado para esse teste.

Dois pontos merecem ser esclarecidos antes da apresentação dos resultados do teste. Um deles é que a extração de iso-superfície com valor 0 não funcionou com o vtkImplicitModeller, então com o esse filtro foi utilizado o valor 0.01. O segundo ponto é que, para manter a igualdade de condições no teste, a partir das dimensões do volume de saída obtidas pela aplicação do `vtkSamplingFromContoursFilter` com uma dada resolução de amostragem, era estabelecida a dimensão de amostragem (número de pontos amostrados nas direções x, y e z) para o vtkImplicitModeller.

A tabela 6.6 resume os resultados obtidos nas várias configurações do teste.

(1) Resolução	(2) Volume de Saída (Dimensão)	(3) Nº de Polígonos Gerados vtkImplicitModeller vtkSamplingFromContoursFilter	(4) Proporção do Nº de Polígonos vtkImp... / vtkSamp...	(5) Tempo Médio (seg) vtkImplicitModeller vtkSamplingFromContoursFilter	(6) Proporção do Tempo vtkImp... / vtkSamp...
10x10	122 x 77 x 22 = 206668	43090 47488	0,91	1703,44 9,65	176,52
15x15	82 x 52 x 22 = 93808	22734 24985	0,91	764,67 5,22	146,49
20x20	62 x 39 x 22 = 53196	14640 16290	0,90	431,93 3,71	116,42

Tabela 6.6 Resultados dos testes envolvendo vtkImplicitModeller e `vtkSamplingFromContoursFilter`.

As colunas 1 e 2 apresentam a resolução utilizada e o correspondente volume obtido, respectivamente. Na coluna 3 estão os valores, discriminados para cada um dos filtros, do número de polígonos que compõem a superfície gerada. Da mesma forma, na coluna 5 é apresentado o tempo médio, obtido a partir da média aritmética de 3 medidas. Os números das colunas 4 e 6 representam a proporção dos valores presentes na colunas 3 e 5, respectivamente, para cada uma das resoluções utilizadas.

Pela coluna 4, vê-se que o quantidade de polígonos em uma superfície gerada usando o vtkImplicitModeller foi sempre menor, mantendo uma proporção muito próxima uma da outra, o que permitiria um *rendering* mais rápido. Entretanto, o tempo gasto na geração da superfície é muitas vezes maior, conforme os valores da coluna 6. Um tempo maior no teste com o vtkImplicitModeller, está de acordo com as expectativas, visto que o passo do cálculo de distância nesse filtro é realizado para todos os pontos do volume, e em relação a todos os contornos no volume e não somente àqueles da mesma fatia do ponto. Apesar disso, a diferença de tempo surpreendeu.

Algumas comparações dos valores presentes na tabela 6.6 estão apresentados na tabela 6.7.

(1) Relação	(2) Proporção da Dimensão do Volume de Saída	(3) Proporção do N° de Polígonos Gerados vtkImplicitModeller vtkSamplingFromContoursFilter	(4) Proporção do Tempo Médio vtkImplicitModeller vtkSamplingFromContoursFilter
(10x10) / (15x15)	2,20	1,90	2,23
		1,90	1,85
(10x10) / (20x20)	3,89	2,94	3,94
		2,92	2,60
(15x15) / (20x20)	1,76	1,55	1,77
		1,53	1,41

Tabela 6.7 Comparação entre valores presentes na tabela 6.6.

A coluna 1 expressa quais dados (linhas) da tabela 6.6 estão sendo relacionados. O número em cada linha da coluna 2 é a proporção entre o volume obtido pelo uso de resoluções diferentes. Os filtros são considerados separadamente nas colunas 3 e 4, comparando-se o número de polígonos gerados e o tempo médio, respectivamente, em relação à variação na resolução de amostragem.

Comparando os valores da coluna 2 com os da coluna 4 referentes ao vtkImplicitModeller, percebe-se que a proporção no aumento do volume de saída reflete, de forma aproximadamente linear, na proporção entre os tempos, uma vez que os pontos adicionados, nesse caso, são considerados no processamento. Já com as proporções referentes ao vtkSamplingFromContoursFilter o efeito é diferente. Pode-se ver que os valores da coluna 4 são inferiores ao da coluna 2. Esse resultado está compatível e reforça aquele obtido com o conjunto

de dados 2 (ver tabela 6.3). A coluna 3 mostra que, para cada filtro individualmente, a proporção entre o número de polígonos gerados variando-se a resolução de amostragem, tem valores muito próximos; em outras palavras, uma variação na resolução provoca o mesmo aumento (ou diminuição) no número de polígonos proporcional, tanto no caso do vtkImplicitModeller como no caso do vtkSamplingFromContoursFilter.

A figura 6.6 mostra alguns resultados do *rendering*, usando cada um dos filtros, de acordo com a resolução de amostragem.

Figura 6.6 Reconstrução empregando vtkSamplingFromContoursFilter (a,b,c) e vtkImplicitModeller (d,e,f).

(a) e (d) Resolução 20 x 20.

(b) e (e) Resolução 15 x 15.

(c) e (f) Resolução 10 x 10.

Pela inspeção das imagens na figura 6.6, observa-se que uma característica predominante é a suavidade nas superfícies reconstruídas com vtkSamplingFromContoursFilter, enquanto nas obtidas com o vtkImplicitModeller têm regiões com aspecto de escada. Uma outra observação é que o canal, apontado pelas setas, está ausente nas imagens inferiores da figura 6.6; fato que, provavelmente, está relacionado com a forma de implementação do filtro vtkImplicitModeller.

Em todos os testes realizados foi empregado um objeto do tipo vtkCleanPolyData, que trata casos de duplicações e degenerações, para verificar se era possível diminuir o número de

polígonos gerados. Os resultados da aplicação desse objeto não acusaram nenhuma diferença na quantidade de polígonos, indicando que as triangulações na criação dos modelos tridimensionais foram bem comportadas.

Nos testes realizados foram gerados arquivos no formato VRML. Algumas diferenças entre as imagens vistas no ambiente do VTK com as vistas no ambiente do VRML foram notadas. Isso está relatado na próxima subseção.

6.1.4 Imagens no Formato do VRML

Utilizando um objeto da classe `vtkVRMLExporter`, foram produzidos arquivos no formato do VRML para posterior manipulação através de um *browser* adequado. As imagens no VTK e no VRML são semelhantes de determinados pontos de vista, como demonstra a imagem gerada com o conjunto de dados 1 apresentada na figura 6.7(a). Entretanto, algumas diferenças foram percebidas, as quais estão, também, ilustradas na figura 6.7.

Figura 6.7 Detalhes de Imagens vistas no ambiente do VRML (a,b,c,d) e do VTK (e).
(a) Proveniente do Conjunto de dados 1.
(b) e (c) Proveniente do Conjunto de dados 2.
(d) e (e) Proveniente do Conjunto de dados 3.

Na figura 6.7(b) a seta aponta para um detalhe que não aparece no canal quando visto no VTK. Esse detalhe desaparece (figura 6.7(c)) quando a reconstrução é feita com os dados após a correção geométrica, apesar da imagem ser diferente da vista no VTK (fundo escuro no interior do dente). A figura 6.7(e) mostra detalhe de uma imagem vista no VTK, que se apresenta diferente quando vista no VRML (figura 6.7(d)). Nessa figura, a elipse destaca uma superfície aberta em uma e fechada na outra imagem, e o retângulo delimita uma região onde algumas estruturas estão presentes na imagem da direita e ausentes na da esquerda. As fontes dessas diferenças podem ser ou o próprio filtro `vtkVRMLExporter`, ou um problema na determinação de

quais faces dos polígonos das superfícies são internas e quais são externas, oriundo da natureza local do algoritmo *Marching Cubes*.

Nas reconstruções feitas nos testes, as superfícies obtidas com o `vtkSamplingFromContoursFilter` apresentam regiões que quebram a suavidade (a região marcada por uma elipse na figura 6.6(a), por exemplo). A Seção subsequente analisa essa característica e outros aspectos de caráter mais geral.

6.2 Considerações Gerais

Nas subseções da Seção 6.1 foram apresentados e analisados os resultados dos testes realizados. Nesta Seção são tratadas características gerais do processo de reconstrução usando o filtro implementado neste projeto de mestrado, o `vtkSamplingFromContoursFilter`.

Nos testes realizados observou-se que algumas regiões das superfícies reconstruídas apresentavam uma quebra na continuidade, criando uma variação brusca na forma nessas regiões. Um exemplo dessa falta de suavidade está indicado na figura 6.6(a) por uma elipse, e pode ser percebida, também, nas figuras referentes a outros testes.

Essa característica de descontinuidade está relacionada com três fatores:

1. Modo de cálculo das distâncias;
2. Formato dos polígonos que descrevem os contornos, especialmente nos trechos curvos;
3. Posicionamento relativo de contornos relacionados em fatias adjacentes.

Como a distância de um ponto a um determinado contorno (aresta do polígono) foi definida como sendo a distância entre o ponto e a aresta ou entre o ponto e um dos extremos da aresta, dependendo de sua projeção ortogonal (capítulo 5), numa região próxima à junção de duas arestas ocorrerá uma variação nos valores de distância. A figura 6.8 será referenciada para uma explicação mais detalhada.

Os pontos que estão sobre os segmentos de reta mais finos são equidistantes às arestas com as quais existe o paralelismo. Esse é o comportamento dos pontos que se localizam nas regiões A e B. Os pontos na região C têm valores de distância variantes, não apresentando a mesma uniformidade observada nas outras duas regiões, mesmo que estejam no prolongamento das retas que passam pelas regiões A e B. São em regiões semelhantes à C que ocorrem as descontinuidades mencionadas acima, quebrando a suavidade da superfície. Uma diminuição no tamanho de tais regiões possibilita reconstruções com superfícies mais uniformes. Essa redução pode ser obtida fazendo com que a variação de inclinação de uma aresta para outra seja pequena,

isto é, modelando as curvas dos contornos com maior definição.

Figura 6.8 Configuração da junção de duas arestas num polígono.

O posicionamento relativo incorreto, entre contornos relacionados em fatias adjacentes, também pode provocar regiões com quebra de suavidade, pois isso faz com que pontos correspondentes numa fatia e na seguinte fiquem erroneamente localizados no volume final gerado. Esse fato pode ser percebido em imagens da figura 6.5, examinando as figuras 6.5(c) e (d). Nessa última é mostrada uma superfície mais suave, e os contornos sofreram um reposicionamento através de uma correção geométrica. É importante ressaltar que a etapa de registro (obtenção de referenciais ou pontos de controle) deve ser considerada e bem planejada para se obter sucesso maior numa reconstrução.

Um outro aspecto do processo de reconstrução por esse tipo de abordagem (amostragem por distância à geometria), diz respeito à extração de contornos, numa mesma fatia, que apresentem alguma proximidade. Na figura 6.5, observa-se que em algumas partes da junção da coroa com a raiz ocorre uma interposição de superfícies, que nesse caso não é correto. Essa invasão de uma superfície pela outra pode ter origem em duas fontes:

1. Interseção durante o processo de extração de contornos;
2. Cruzamento de arestas durante a execução do traçado da superfície.

O caso 1 acima pode ser contornado instruindo o operador a deixar uma distância de dois pixels (ou duas coordenadas inteiras) entre contornos vizinhos. O caso 2 também é melhorado por essa atitude, podendo também ser controlado pelo aumento da resolução ou, em um passo de aprimoramento do algoritmo utilizado na implementação, adicionando a ele uma estratégia de subdivisão adaptativa nas regiões consideradas "críticas". O fechamento de contornos também pode causar a interseção de superfícies, visto que a estratégia adotada para o fechamento é a criação de fatias adicionais.

A forma de implementação do algoritmo de preenchimento de regiões também permite que contornos descrevendo buracos sejam corretamente mapeados, como foi feito com o conjunto de dados 3 gerando o canal (figuras 6.6(a), (b) e (c)).

Um outro ponto que pôde ser observado pelos resultados obtidos nos teste é que, tanto para cortes verticais (conjuntos de dados 1 e 2) como para horizontais (conjunto 3), as superfícies são geradas e o dente virtual reconstruído, com recuperação de forma bastante satisfatória.

Os dados enviados pelos colaboradores da Inglaterra foram gerados pelo editor de contornos desenvolvido por eles [Goe 98]. Esses dados foram processados, por eles, utilizando o software NUAGES (<http://www.inria.fr/prisme/personnel/geiger/nuages.html>) de reconstrução de imagens médicas. Nesse caso, foi observado um problema de inconsistência na reconstrução em fatias mais baixas do volume. Resultados preliminares não acusaram que esse tipo de problema ocorre com a utilização do filtro implementado nesta dissertação. Um prosseguimento do projeto do dente virtual será a comparação dos resultados obtidos neste trabalho de mestrado com aqueles obtidos pelos colaboradores ingleses [Goe 98].

Nesta Seção foram colocadas observações e feitas avaliações de pontos importantes característicos do algoritmo implementado, bem como do processo de reconstrução 3D de dentes no qual ele está inserido. A Seção seguinte apresenta conclusões pertinentes ao projeto e provê algumas sugestões para extensão e melhoria do processo de reconstrução.

6.3 Conclusões

Pelos resultados obtidos nos testes, conclui-se que o algoritmo implementado no filtro `vtkSamplingFromContoursFilter` é aplicável, tanto em desempenho quanto no aspecto visual, ao processo de geração de um modelo 3D de dentes a partir de um conjunto de contornos planares, através da reconstrução de superfícies representando as várias partes que formam um dente, com boa recuperação de forma, como pode ser percebido na figura 6.9.

Figura 6.9 Modelo 3D de dente reconstruído usando o `vtkSamplingFromContoursFilter`

O modelo 3D do dente da figura 6.9 é composta por duas estruturas, uma interna e outra externa. A superfície da estrutura interna foi construída a partir de 2387 pontos formando 157 polígonos em 64 fatias, e a superfície da estrutura externa a partir de 4017 pontos, 133 polígonos e 81 fatias.

Para ampliar a aplicabilidade, melhorar o desempenho e o produto visual do algoritmo implementado, algumas extensões e generalizações poderiam ser imediatamente incorporadas a ele, a saber:

1. Manipulação de coordenadas no formato real, através de uma adaptação no algoritmo de preenchimento de regiões e cálculo automático do *aspect ratio*;
2. Alterar o parâmetro de dimensão de amostragem, fornecendo o número de pontos nas direções x e y ao invés de estabelecer de quantos em quantos pontos (resolução) fazer uma amostragem;
3. Diminuição do volume de saída, pela geração de um volume de saída que englobe somente a região de interesse, e pelo cálculo automático das coordenadas da origem desse volume para um correto posicionamento;
4. Criação de fatias intermediárias, para o caso da distância entre fatias não ser constante. É necessário que mais pontos tenham a distância calculada.

O processo de obtenção de dados por fatiamento do dente é bastante demorado, além de ser uma provável fonte de distorções. Atualmente, está disponível a microtomografia, que permite obter os dados de uma forma mais rápida e precisa (<http://www.lab3d.odont.ku.dk/main-docs/Projects/MicroCT/>). Soma-se a essas vantagens o fato do processo ser não destrutivo (<http://www.skyscan.com/CMTINF.HTM>), permitindo que um único dente sirva para vários testes.

Em função dos objetivos de mais longo prazo, neste trabalho foi considerada a utilização de dados provenientes de dentes reais ao invés de dados modelados artificialmente. Dados reais são mais fiéis a estrutura do objeto sob análise, o que é essencial, também, para aplicações em educação e treinamento de profissionais.

Os tempos de processamento e os resultados visuais foram conseguidos usando um conjunto de dados pequeno e um equipamento de configuração modesta para os dias atuais, mesmo para o segmento doméstico. Isso permite vislumbrar a viabilidade de uso da abordagem implementada para aplicação em educação, treinamento e demonstração. Deve-se considerar, ainda, o baixo custo do VTK, a facilidade de criação de interfaces (como a criada para o uso do filtro, exposta no capítulo 5) e a interação do VRML e do próprio VTK, todas essas ferramentas desejáveis para as diversas aplicações em potencial.

Os algoritmos aqui expostos representam ferramentas fundamentais para a criação de um procedimento básico de reconstrução, sobre o qual muitas operações e novas técnicas podem ser aplicadas, incluindo técnicas de interface e exploração. Este trabalho, fornece assim, o primeiro e necessário passo na direção de um sistema completo de educação e demonstração de estruturas e procedimentos odontológicos, podendo ser estendido a médio prazo para incluir ferramentas de treinamento e diagnóstico.

REFERÊNCIAS BIBLIOGRÁFICAS

- [Alg 96] Algorri, M.; Schmitt, F. - "Surface Reconstruction from Unstructured 3D Data", **Computer Graphics Forum**, vol. 15, no. 1, 1996, pp. 47-60.
- [Baj 95] Bajura, M.; Neumann, U. - "Dynamic Registration Correction in Video-Based Augmented Reality Systems", **IEEE Computer Graphics and Applications**, vol. 15, no. 5, September 1995, pp. 52-60.
- [Ben 79] Bentley, J. L.; Ottmann, T. A. - "Algorithms for Reporting and Counting Geometric Intersections", **IEEE Transactions on Computers**, vol. C-28, no. 9, September 1979, pp. 643-647.
- [Ble 95] Bleicher, A.; Heiden, W. - WWW Page: **The Responsive Workbench**, last modified 28 July 1995. (<http://viswiz.gmd.de/VMSSD/PAGES.en/projects.workbench.html>)
- [Bow 95] Bowskill, J.; Downie, J. - "Extending The Capabilities of the Human Visual System An Introduction to Enhanced Reality", **ACM SIGGRAPH Computer Graphics**, vol. 29, no. 2, May 1995, pp. 61-65.
- [Bro 92] Brodlie, K. W.; Carpenter, L. A.; Earnshaw, R. A.; Gallop, J. R.; Hubbard, R. J.; Mumford, A. M.; Osland, C. D.; Quarendon, P. (eds.) - **Scientific Visualization - Techniques and Applications**, Springer-Verlag, 1992.
- [Bro 95] Brown, J. R.; Earnshaw, R.; Jern, M.; Vince, J. - **Visualization Using Computer Graphics to Explore Data and Present Information**, John Wiley & Sons, 1995.
- [Bro 96] Bro-Nielsen, M.; Larsen, P.; Kreiborg, S. - "Virtual Teeth: A 3D method for editing and visualizing small structures in CT scans", **Proceedings of Computer Assisted Radiology (CAR'96)**, pp. 921-924. (http://www.imm.dtu.dk/documents/users/mvox/virteeth_submit.ps.gz)
- [Buc 95] Buck, T.A.; Ehrlicke, H.; Strasser, W.; Thurfjell, L. - "3-D Segmentation of Medical Structures by Integration of RayCasting with Anatomic Knowledge", **Computers & Graphics**, vol. 19, no. 3, 1995, pp. 441-449.

- [CG 87] ACM SIGGRAPH Computer Graphics, Special issue on Visualization in Scientific Computing, vol. 21, no. 6, November 1987.
- [Cos 96] Costa, A. R. - Dissertação de Mestrado: **As Implicações Teóricas e a Implementação de um Completo Sistema de Visualização de Imagens Médicas**, ICMSC-USP, 1996.
- [Cra 94] Crawfis, R.; Max, N.; Nieslson, G.; Hansen, C. - **Advanced Techniques for Scientific Visualization**, ACM SIGGRAPH'94 Course Notes.
- [Die 96] Dieberger, A. - WWW Page: **The Information City - A Step towards Merging of Hypertext and Virtual Reality**, last modified 26 October 1996.
(<http://www.lcc.gatech.edu/faculty/dieberger/HT93.poster.writeup.html>)
- [Doy 93] Doyle, M. D.; Ang, C.; Raju, R.; Klein, G.; Williams, B. S.; DeFanti, T.; Goshtasby, A.; Grzeszczuk, R.; Noe, A. - **Processing cross-sectional image data for reconstruction of human developmental anatomy from museum specimens**, 1993.
(<http://www.muritech.com/visembryo/sigbio93/SIGBIO93.html>)
- [Eko 91] Ekoule, A. B.; Peyrin, F. C.; Odet, C. L. - "A Triangulation Algorithm from Arbitrary Shaped Multiple Planar Contours", **ACM Transactions on Graphics**, vol. 10, no. 2, April 1991, pp. 182-199.
- [Elv 92] Elvins, T. T. - "A Survey of Algorithms for Volume Visualization", **ACM SIGGRAPH Computer Graphics**, vol. 26, no. 5, August 1992, pp. 194-201.
- [Elv 94] Elvins, T. T. - "A Survey of Algorithms for Volume Visualization", In SIGGRAPH'94 Course Notes: **Introduction to Volume Visualization: Imaging Multi-Dimensional Scientific Data, Computer Graphics**.
- [Fer 96] Ferreira, R. A. - "Odontologia em Imagens", **Revista da APCD**, vol. 50, no. 3, maio/junho 1996, pp. 218-228.
- [Fol 96] Foley, J. D.; van Dam, A.; Feiner, S. K.; Hughes, J. F. - **Computer Graphics Principles and Practice**, Second Edition in C, Addison-Wesley, 1996.
- [Goe 98] Goetz, M. R.; Day, A. M. - "Surface Reconstruction for Teeth", **Proceedings of EUROGRAPHICS UK**, 16th Annual Conference, 25-27 March, Leeds, UK, 1998.
- [Gon 87] Gonzalez, R. C.; Wintz, P. - **Digital Image Processing**, Second Edition, Addison-Wesley, 1987.
- [Gri 95] Grinstein, G.; Kienegger-Domik, G.; Owen, G. S. - **Guidelines for Teaching a Course in Visualization**, Tutorial 4 Visualization'95.
- [Jon 94] Jones, M. W.; Chen, M. - "A New Approach to the Construction of Surfaces from Contour Data", **Proceedings of the EUROGRAPHICS'94**, vol. 13, no. 3, 1994, pp. 75-84.
- [Kau 90] Kaufmann, A. - **3D Volume Visualization**, Tutorial Note 12 Eurographics'90, EG 90 TN 12.
- [Kau 93] Kaufman, A.; Cohen, D.; Yagel, R. - "Volume Graphics", **IEEE Computer**, vol. 26, no. 7, July 1993, pp.51-64.
- [Kel 93] Keller, P. K.; Keller, M. M. - **Visual Cues: Practical Data Visualization**, IEEE Computer Society Press and IEEE Press, 1993.

- [Kre 96] Kreiborg, S.; Larsen, P.; Bro-Nielsen, M.; Darvann, T. - "A 3-dimensional analysis of tooth formation and eruption in a case of Apert syndrome", **Proceedings of Computer Assisted Radiology (CAR'96)**, pp. 1066-1068.
(<http://orkide.lab3d.odont.ku.dk/main-docs/Publications/Pub-1996/TeethCAR96/TeethCAR96.html>)
- [Lai 94] Laidlaw, D. H.; Fleischer, K. W.; Barr, A. H. - "Classification of Material Mixtures in Volume Data for Visualization and Modeling", **Technical Report CS-TR-94-07**, Computer Science Department, California Institute of Technology, 1994.
(<ftp://ftp.cs.caltech.edu/tr/cs-tr-94-07.ps.Z>)
- [Lap 83] Lapa, N.; Cavallante, S. L. - **Matemática Vol. 3**, Editora Moderna, 1983.
- [Lee 81] Lee, D. T.; Drysdale, R. L. - "Generalization of Voronoi Diagrams in the Plane", **SIAM Journal of Computing**, vol. 10, no. 1, February 1981, pp. 73-87.
- [Lee 82] Lee, D. T. - "Medial Axis Transformation of a Planar Shape", **IEEE Transactions on Pattern Analysis and Machine Intelligence**, vol. PAMI-4, no. 4, July 1982, pp. 363-369.
- [Lor 87] Lorensen, W. E.; Cline, H. E. - "Marching Cubes: A high resolution 3D surface construction algorithm", **ACM SIGGRAPH Computer Graphics**, vol. 21, 1987, pp.163-169.
- [Lor 95] Lorensen, W. E. - "Marching Through the Visible Man", **Proceedings of the IEEE Visualization'95**, pp. 368-373.
- [Mat 94] Matveyev, S. B. - "Approximation of Isosurface in the Marching Cube: Ambiguity Problem", **Proceedings of the IEEE Visualization'94**, pp. 288-292.
- [Mer 79] Merchant, D. C. - **Notas de Aula: Analytical Photogrammetry Theory and Practice - Part I**, Department of Geodetic Science, The Ohio State University, USA, 1979.
- [Mey 92] Meyers, D.; Skinner, S.; Sloan, K. - "Surface from Contours", **ACM Transactions on Graphics**, vol. 11, no. 3, July 1992, pp. 228-258.
- [Min 95] Minghim, R. - **PhD Thesis: On Sound Support for Visualization**, School of Information Systems, University of East Anglia, UK, March 1995.
- [Min 97] Minghim, R.; Oliveira, M. C. F. - "Uma Introdução à Visualização Computacional", **Jornadas de Atualização em Informática (JAI-97)**, XVII Congresso da SBC, Brasília, 2 a 8 de agosto de 1997, 45p.
- [Mon 94] Montani, C.; Scateni, R.; Scopigno, R. - "Discretized Marching Cubes", **Proceedings of the IEEE Visualization'94**, pp. 281-287.
- [Mon 94a] Montgomery, K.; Muriel, D. R. - **Improvements in semiautomated serial-section reconstruction and visualization of neural tissue from TEM images**, 1994.
(http://biocomp.arc.nasa.gov/papers/spic_paper.94)
- [Mys 95] Myszkowski, K.; Okuneva, G.; Herder, J.; Kunii, T. L.; Ibusuki, M. - "Visual Simulation of the Chewing Process for Dentistry", **Proceedings of the British Computer Society Workshop on Visualization and Modelling**, Leeds, UK, December 1995, pp. 1-26.
- [Non 98] Nonato, L. G. - **Tese de Doutorado: Reconstrução Volumétrica de Objetos Dados por Seções Bidimensionais**, Departamento de Matemática, PUC-RJ, 1998.
(com defesa prevista para 19 de junho de 1998)

- [Oli 96] Oliva, J.; Perrin, M.; Coquillart, S. - "3D Reconstruction of Complex Polyhedral Shapes from Contours using a Simplified Generalized Voronoï Diagram", **Proceedings of the EUROGRAPHICS'96**, vol. 15, no. 3, 1996, pp. 397-408.
- [ORo 93] O'Rourke, J. - **Computational Geometry in C**, Cambridge University Press, 1993.
- [Owe 95] Owen, G. S. - Course in WWW: **HyperVis - Teaching Scientific Visualization Using Hypermedia**, 1995.
(<http://ironduke.cs.gsu.edu/~matgso/hypervis>)
- [Pra 91] Pratt, W. K. - **Digital Image Processing**, Second Edition, John Wiley & Sons, 1991.
- [Pra 96] Prado, A. H. M. A. - Dissertação de Mestrado: **Interpolação de Imagens Tomográficas Através de Matching**, ICMSC-USP, 1996.
- [Pre 85] Preparata, F. P.; Shamos, M. I. - **Computational Geometry An Introduction**, Springer-Verlag, 1985.
- [Pre 94] Preece, J.; Rogers, Y.; Sharp, H.; Benyon, D.; Holland, S.; Carey, T. - **Human-Computer Interaction**, Addison-Wesley, 1994.
- [Ros 95] Rosenblum, L. J.; Bryson, S.; Feiner, S. K. - "Virtual Reality Unbound", **IEEE Computer Graphics and Applications**, vol. 15, no. 5, September 1995, pp. 19-21.
- [Rou 94] Roux, C.; Coatrieux, J. L. (co-chairs); Dillenseger, J.; Fishman, E. K.; Loew, M.; Meinzer, H.; Pearlman, J. D. (panelists) - Panel: "Vizualization in Medicine: Virtual Reality or Actual Reality?", **Proceedings of the IEEE Visualization'94**, pp. 396-399.
- [Rum 91] Rumbaugh, J.; Blaha, M. - **Object-Oriented Modeling and Design**, Prentice-Hall, 1991.
- [Sak 94] Sakas, G.; Schreyer, L.; Grimm, M. - "Case Study: Visualization of 3D Ultrasonic Data", **Proceedings of the IEEE Visualization'94**, pp. 369-373.
- [Sch 96] Schroeder, W.; Martin, K.; Lorensen, B. - **The Visualization Toolkit - An Object-Oriented Approach to 3D Graphics**, Prentice-Hall, 1996.
- [Sch 96a] Schenberg, M. - "Pesquisadora cria dente virtual", **Jornal do Campus**, ano 15, no. 177, outubro 1996, p. 5.
- [Sch 97] Schroeder, W. J.; Martin, K.; Lorensen, W. - **The Visualization Toolkit - An Object-Oriented Approach to 3D Graphics**, 2nd Edition, Prentice-Hall, 1997.
- [Sei 95] Seipel, S.; Wagner, I.; Koch, S.; Scheneider, W. - "Three-dimensional visualization of the mandible: A new method for presenting the periodontal status and diseases", **Comput. Meth. Programs Biomed** 46, 1995, pp. 51-57.
- [Sen 90] Senay, H. (organizer); Feiner, S.; Fisher, S. S.; Grinstein, G.; MacKinlay, J. D.; Treinish, L. A. (panelists) - Panel: "Interaction Issues in Visualization: Requirements, Techniques, and Devices", **Proceedings of the IEEE Visualization'90**, pp. 393-400.
- [Shi 97] Shimabukuro, M. H. - Qualificação de Mestrado: **Dentes Virtuais: Visualização para Apoio a Atividades Odontológicas**, Departamento de Ciências de Computação e Estatística, ICMC-USP, São Carlos, abril 1997.

- [Shi 98a] Shimabukuro, M. H.; Minghim, R.; Licciardi, P. R. D. B. - "Implementing Visualization Procedures in Dental Applications", **Proceedings of SCT'98 - World Multiconference on Systemics Cybernetics and Informatics and the International Conference on Information Systems Analysis and Synthesis**, Orlando, USA, July 1998.
(a ser publicado)
- [Shi 98b] Shimabukuro, M. H.; Minghim, R.; Licciardi, P. R. D. B. - "Visualisation and Reconstruction in Dentistry", **Proceedings of Information Visualisation'98 (IV'98)**, London, UK, July/August 1998.
(a ser publicado pela IEEE Computer Society Press)
- [Sta 96] State, A.; Hirota, G.; Chen, D. T.; Garrett, W. F.; Livingston, M. A. - "Superior Augmented Reality Registration by Integrating Landmark Tracking and Magnetic Tracking", **Proceedings of the SIGGRAPH Computer Graphics'96**, pp. 429-438
- [Tom 84] Tommaselli, A. M. G.; Chaparro, B. - Notas de Aula: **Transformações de Coordenadas**, Curso de Pós-Graduação em Ciências Geodésicas, UFPR, Curitiba, 1984.
- [Tra 95] Traina, A. J. M.; Oliveira, M. C. F. - Notas de Aula: **Computação Gráfica**, Curso de Pós-Graduação, Departamento de Ciências de Computação e Estatística, ICMC-USP, São Carlos, 1995.
- [Tre 96] Treinish, L.; Silver, D. - "Visualizing the Visible Human", **IEEE Computer Graphics and Applications**, vol. 16, no.1, January 1996, pp.7-9.
- [Udu 91] Udupa, J. K.; Herman, G. T. - **3D Imaging in Medicine**, CRC Press, 1991.
- [Wat 92] Watt, A.; Watt, M. - **Advanced Animation and Rendering Techniques - Theory and Practice**, ACM Press, 1992.
- [Wol 93] Wolf, R. S.; Yaeger, L. - **Visualization of Natural Phenomena** - Springer-Verlag, 1993.

APÊNDICE A

Página do Manual VTK para o Filtro **vtkSamplingFromContoursFilter**

NOME

vtkSamplingFromContoursFilter - gera uma malha 3D com escalares representando distâncias a uma geometria de entrada definida por contornos planares.

RESUMO

```
#include "vtkSamplingFromContoursFilter.h"
```

```
class vtkSamplingFromContoursFilter : public vtkDataSetToStructuredPointsFilter
```

```
vtkSamplingFromContoursFilter();
```

```
char *GetClassName() { return "vtkSamplingFromContoursFilter"; };
```

```
void SetSliceSamplingResolution (int srx, int sry) ;
```

```
void SetOutputAspectRatio (float arx, float ary, float arz) ;
```

```
void SetOutputOrigin (float ox, float oy, float oz) ;
```

```
void CloseFirstSurfaceOn( ) ;
```

```
void CloseFirstSurfaceOff( ) ;
```

```
void CloseLastSurfaceOn( ) ;
```

```
void CloseLastSurfaceOff( ) ;
```

```
void SetCloseFirstSurfaceZero( float f0p) ;
```

```
void SetCloseLastSurfaceZero( float l0p) ;
```

DESCRIÇÃO

`vtkSamplingFromContoursFilter` faz uma amostragem de pontos num volume de entrada formado por um conjunto de contornos em fatias paralelas. Para cada ponto, é calculada a distância em relação à aresta mais próxima, considerando os contornos que estão presentes na mesma fatia do ponto em questão.

SUMÁRIO

`void SetSliceSamplingResolution (int srx, int sry)`

Define a resolução de amostragem (intervalo entre pontos) para cada uma das fatias. O valor relativo à direção z é sempre 1.

`void SetOutputAspectRatio (float arx, float ary, float arz)`

Estabelece a razão de aspecto do volume de saída. Essa informação não é extraída dos dados de entrada.

`void SetOutputOrigin (float ox, float oy, float oz)`

Determina as coordenadas da origem do volume de saída. Utilizado para posicionar corretamente as superfícies. Essa informação não é extraída dos dados de entrada.

`void CloseFirstSurfaceOn( ) [void CloseLastSurfaceOn( )]`

Habilita o fechamento dos contornos presentes na primeira [última] fatia do conjunto.

`void CloseFirstSurfaceOff( ) [void CloseLastSurfaceOff( )]`

Inibe o fechamento dos contornos presentes na primeira [última] fatia do conjunto.

`void SetCloseFirstSurfaceZero( float f0p) [void SetCloseLastSurfaceZero( float l0p)]`

Indica, através de um valor no intervalo aberto (0,1), a posição do ponto mais alto da superfície de fechamento da primeira [última] fatia do conjunto. Quanto menor o valor, mais próxima ao contorno original ocorrerá o fechamento.

`vtkSamplingFromContoursFilter()`

Assume os seguintes valores padrão para os parâmetros: resolução de amostragem = (1,1), razão de aspecto = (1,1,1), origem = (0,0,0), fechamento inibido (se habilitado, posição do ponto mais alto da superfície de fechamento = 0.5).

APÊNDICE B

Formato dos Dados de Entrada para `vtkSamplingFromContoursFilter`

O formato de dados do VTK utilizado como entrada para o filtro `vtkSamplingFromContoursFilter` é o `POLYGONAL DATASET`, conforme o exposto no capítulo 5. Nesse formato de dados são definidas as seções `POINTS`, `VERTICES`, `LINES`, `POLYGONS` e `TRIANGLE_STRIP`s. Na seção `POINTS` são especificadas as coordenadas dos pontos que compõem o *dataset*. As demais seções são utilizadas para definir a topologia do *dataset* [Sch 96]. Para o caso da especificação dos contornos dos dentes, isto é, para descrever os polígonos que representam os contornos em cada fatia, as seções `POINTS` e `POLYGONS` são suficientes. A figura B.1 dá um exemplo da especificação de um arquivo de entrada válido para o `vtkSamplingFromContoursFilter`, criado pelo programa complementar de geração de arquivo no formato VTK desenvolvido neste trabalho.

No trecho do arquivo apresentado na figura B.1, são mostrados o cabeçalho e as seções `POINTS` e `POLYGONS`. O cabeçalho é composto por uma identificação de versão, seguido de um comentário livre, e a informação de que se trata de um `DATASET POLYDATA` especificado no formato ASCII. Juntamente com a palavra-chave `POINTS` são fornecidos a quantidade de pontos a ser especificada (1604) e o formato das coordenadas (float). Logo abaixo, é fornecida a lista de coordenadas dos pontos, e para cada ponto é associado, implicitamente, um índice de acordo com a sua posição nessa lista (0,1,2...1603). Esse índice é utilizado para definir os vértices dos polígonos na seção `POLYGONS`. Na figura B.1, estão mostrados dois trechos de especificação de coordenadas, um para a fatia 0 e outro para a fatia 5. Na seção seguinte, além do identificador `POLYGONS` são passados dois valores, um para o número de polígonos (25) e outro para informar qual é o espaço necessário para armazenar a lista de polígonos, ou seja, a quantidade de pontos somado a uma posição para cada polígono ($1604 + 25 = 1629$). No exemplo da figura

B.1 são mostrados trechos da definição de três polígonos, com o primeiro número indicando quantos vértices o respectivo polígono possui, seguido da lista de índices dos vértices.

```
# vtk DataFile Version 1.0
File built by vtk-ASCII file generation program
ASCII
DATASET POLYDATA
POINTS 1604 float
310.000000 298.000000 0.000000
308.000000 295.000000 0.000000
306.000000 291.000000 0.000000
...
340.000000 173.000000 5.000000
340.000000 177.000000 5.000000
342.000000 186.000000 5.000000
...
...
POLYGONS 25 1629
27 0 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 26
...
17 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441
...
68 1536 1537 1538 1539 1540 1541 ...
```

Figura B.1 Exemplo de um *dataset* válido para o `vtkSamplingFromContoursFilter`.

Na atual versão do filtro, algumas características devem ser observadas:

1. Os valores para as coordenadas dos vértices dos polígonos devem ser inteiros positivos, mesmo que especificados como float;
2. Os valores de *z* não são utilizados para estimar a distância entre as fatias, que é supostamente constante. A correta relação entre as distâncias em *x*, *y* e *z* é estabelecida pela razão de aspecto do volume de saída;
3. Cada arquivo descreve apenas uma superfície, ou seja, no caso da reconstrução envolver várias superfícies, cada uma deve ser especificada em um arquivo separado;
4. Contornos na mesma fatia devem possuir o mesmo valor para a coordenada *z*;
5. Os polígonos devem ser definidos ordenadamente, ou seja, na mesma sequência das fatias.