Thiago Ramos dos Santos

Representação, Visualização e Manipulação de Dados Médicos Tridimensionais

Um Estudo Sobre as Bases da Simulação Cirúrgica Imersiva

Florianópolis - SC - Brasil

2006

Programa de Pós-Graduação em Ciências da Computação
Universidade Federal de Santa Catarina

Thiago Ramos dos Santos

Representação, Visualização e Manipulação de Dados Médicos Tridimensionais

Um Estudo Sobre as Bases da Simulação Cirúrgica Imersiva

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciências da Computação.

Aldo von Wangenheim, Prof. Dr. rer. nat
Orientador

Florianópolis - SC - Brasil

2006

Representação, Visualização e Manipulação de Dados Médicos Tridimensionais

Um Estudo Sobre as Bases da Simulação Cirúrgica Imersiva

Thiago Ramos dos Santos

Dissertação de Mestrado realizada como requisito para a obtenção do título de Mestre em Ciências da Computação dado pelo Programa de Pós-Graduação em Ciências da Computação (PPGCC), do Departamento de Informática e Estatística (INE) da Universidade Federal de Santa Catarina (UFSC), defendida no dia 30 de março de 2006, em Florianópolis, SC, Brasil.

Raul Sidnei Wazlawick, Prof. Dr.
Coordenador do Curso

Banca Examinadora

Aldo von Wangenheim, Prof. Dr. rer. nat
Orientador

Agma Juci Machado Traina, Profa. Dra.
Universidade de São Paulo (USP) - São Carlos/SP

Eros Comunello, Prof. Dr. rer. nat.
Universidade do Vale do Itajaí (UNIVALI) - São José/SC

Luciana Porcher Nedel, Profa. Dra.
Universidade Federal do Rio Grande do Sul (UFRGS) - Porto Alegre/RS

Dedicatória

Aos meus pais, Euclydes e Maristela, sempre apoiando e ajudando em todas as empreitadas.

À Iara, minha namorada, que teve peciência de me aguentar quando não havia tempo para mais nada, a não ser para terminar o mestrado.

Aos meus avós (Vilma, Euclydes, Acedilte e Reynaldo), tios (Bia, Dinho, Rosina, Bebeto e Miriam) e irmãs (Juliana e Joana), sempre torcendo, mesmo que seja de forma distante (alguns muito mais distante).

Ao Rodolfo, meu tio, que, além de fornecer um lar, ainda atua como grande companheiro nas festas, bebidas e afins.

Aos amigos, tanto em Florianópolis quanto no Rio Grande do Sul, em especial ao Pelotas e Petrúcio.

Agradecimentos

Ao prof. Aldo, por estar sempre disposto a uma discussão e a fornecer material para fomentar discussões.

Ao Tiago e Diego, pelo apoio, idéias, discussões e por botar a mão-na-massa quando necessário.

Resumo

Dados tridimensionais referentes a pacientes são utilizados em diversos setores médico-hospitalares, fornecendo embasamento à diagnósticos e orientação durante procedimentos cirúrgicos. No entanto, apesar de bastante úteis estes dados são bastante inflexíveis, não permitindo que o usuário interaja com estes ou os manipule. O emprego de técnicas de computação gráfica e realidade virtual para a representação destes dados sanaria estas dificuldades, gerando representações indivíduais e adaptadas para cada paciente e permitindo a realização de planejamentos cirúrgicos e cirurgias auxiliadas por computador, dentre outras possibilidades. A representação destes dados e as formas de manipulação devem conter um conjunto de elementos e obedecer alguns requisitos para que se obtenha realismo nas aplicações, caso contrário, o emprego destas técnicas não traria grandes vantagens. Analisando os elementos e requisitos a serem obedecidos, é construído um grafo de dependências que mostra as técnicas e estruturas computacionais necessárias para a obtenção de ambientes virtuais imersivos realistas. Tal grafo demonstra as estruturas de dados para representação de sólidos como peça chave para este tipo de aplicativos. Para suprir as necessidades destes, é apresentada uma estrutura de dado capaz de representar uma vasta classe de topologias espaciais, além de permitir rápido acesso a elementos e suas vizinhanças, bem como métodos para a construção de tal estrutura. É apresentada, também, uma aplicação para mensuração de artérias utilizando a estrutura e os métodos previamente mencionados e os resultados por obtidos por estes.

Abstract

Patient-acquired three-dimensional data is widely used in medical environments, suplying base for diagnosis and orientation during surgical procedures. However, though very useful, those kind of data is very inflexible, disallowing the user to interact with them or manipulates it. The use of computer graphics and virtual reality techniques for the representation of those kind of data overcomes those difficulties, generating individual and adapted representations for each single patient and allowing the performance of surgical planings and computer-auxiliated surgeries, among other possibilities. The representation of those data and the manipulation forms must contain a set of elements and obey some requisites in order to obtain realism in the applications, otherwise the employment of those techniques would not bring great advantages. Through the analysis of those elements and requisites, a dependency graph is built which shows the techniques and computational structures required for the obtention of immersive and realistic virtual environments. This graph shows the data structures for solid representation as a central key for those kind of applications. In order to supply the needs of it, a data structure capable of representing a wide class of topologies, and allowing quick access to it's components and their neighbors, is presented, as well as the methods used for it's construction. An application for arterial measurements, which uses those structure and methods, is presented as well as the results obtained with it.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Atualmente, dados tridimensionais referentes aos pacientes são utilizados em diversos setores médico-hospitalares, visando fornecer embasamento aos diagnósticos e orientação durante os procedimentos cirúrgicos. São dados médicos tridimensionais comumente utilizados as séries de Tomografia Computadorizada (CT) e as de Ressonância Magnética (MR). Ambos provêem a capacidade de visualizar, interna e externamente, um determinado volume através de um conjunto de imagens bidimensionais, sendo cada uma destas gerada em uma determinada altura da estrutura em exame (figura 1). Certos tipos de exames possuem também a capacidade de prover informações funcionais do volume em questão. Segundo a *American Heart Association*, estes exames, além de fornecer relevante informação anatômica e funcional, são relativamente não-invasivos e possuem baixos riscos a curto e longo prazo (AMERICAN HEART ASSOCIATION, 2004).

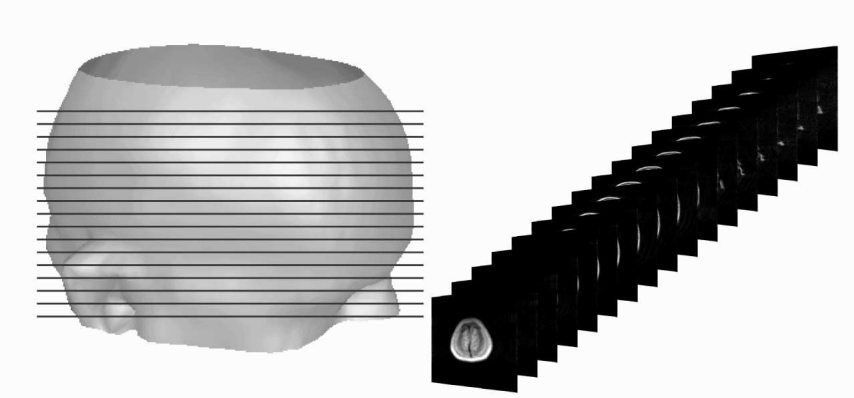

Figura 1: Seqüência de imagens de ressonância magnética possibilitando a visualização interna de uma cabeça humana.

Apesar de bastante utilizadas, a visualização de volumes através de um conjunto de imagens bidimensionais falha nos seguintes aspectos (SANTOS; ABDALA; WANGENHEIM, 2004):

- Existe a perda da noção tridimensional das estruturas representadas pelas imagens;
- É impossível visualizar as estruturas representadas nas imagens por qualquer outro ângulo diferente daquele em que as imagens foram geradas.

Além disso, a manipulação de tais estruturas não é possível, já que isto exige interpretação das imagens e a representação das estruturas através de modelos computacionais geométricos passíveis de manipulação. A manipulação das estruturas presentes em imagens médicas possibilita a realização de planejamentos cirúrgicos e a simulação de cirurgias.

Os problemas inerentes às imagens bidimensionais podem ser resolvidos através da utilização de técnicas de computação gráfica e realidade virtual. Segundo Rosen (1994), realidade virtual é uma tecnologia gerada por computador que permite a visualização de informações em um ambiente simulado, porém, realista. Trabalhos como Scharver et al. (2004b), Kühnapfel et al. (2001), Çakmak, Kühnapfel e Bretthauer (2000) e Robb e Cameron (1994, 1995) demonstram a utilização das técnicas de realidade virtual para solucionar tais problemas. Estes trabalhos demonstram mundos virtuais que permitem ao usuário navegar e interagir em ambientes tridimensionais, gerados por computador, em tempo real. Scharver et al. (2004a) ainda mencionam que o usuário deve ser capaz de sentir os modelos, além de simplesmente vê-los.

Além da solução das inflexibilidades inerentes às imagens bidimensionais, a modelagem de ambientes virtuais através de modelos gráficos computacionais possibilita ainda a exploração de novos rumos na medicina, como a cirurgia por tele-presença (presença virtual) e o tele-diagnóstico (MOLINE, 1997; SHERIDAN, 1992), a tele-manipulação de robôs com finalidade cirúrgica (SATAVA, 1992) e a imersão colaborativa (LEIGH et al., 1999), possibilitando o encontro entre médicos, localizados em pontos remotos, dentro de uma mesma sala de cirurgia virtual. Conforme idealizado por Robb e Cameron (1994), ainda é possível fazer a utilização dos modelos gráficos para o auxílio durante o ato cirúrgico através da mesclagem, dentro de um mesmo ambiente virtual, entre os dados gerados por computador e as estruturas presentes no espaço real (LOK, 2004).

Embora voltados a aplicações diferentes, Pratt, Zyda e Kelleher (1995) afirmam que os sis-

temas que geram ambientes virtuais têm em comum a interação, as imagens tridimensionais e a imersão. A geração de imagens tridimensionais depende da origem dos dados a serem reconstruídos e de sua interpretação. Na aplicação médica, estes dados são constituídos pelo conjunto de imagens representando um determinado volume. Após a interpretação das imagens obtém-se uma representação dos objetos contidos nestas, podendo ser esta uma representação volumétrica por interior, a qual armazena dados de todo o objeto, ou uma representação volumétrica por bordas, onde são armazenados apenas os limites do objeto (SAMET, 1989a). A interação entre usuário e objetos varia conforme a aplicação do ambiente, no entanto, as estruturas de dados que realizam a representação dos objetos, sejam estas por interior ou bordas, devem prover os operadores básicos para a manipulação, tais como os operadores de conjuntos (FOLEY et al., 1990), os operadores de Euler (MÄNTYLÄ, 1988) e os transformadores lineares (conhecidos da álgebra linear clássica). Além disso, pode-se empregar a manipulação dos aspectos visuais, sem modificação das estruturas, tais como a geração de cortes na estrutura através de planos arbitrários e as reconstruções curvilineares (BASTOS et al., 1995; MUSSE et al., 1998) (figura 2). O método para representação computacional das estruturas presentes nas imagens deve ser capaz de portar informações extra (resistência do material, por exemplo), sendo estas utilizadas para modificar o comportamento das estruturas diante das formas de manipulação. Para a imersão, faz-se necessário o uso de equipamentos capazes de gerar sensações imersivas, tais como os HMDs (*Head Mounted Displays*), e táteis, como as luvas capazes de gerar retro-alimentação de força (*force-feedback*) (SHERIDAN, 1992).

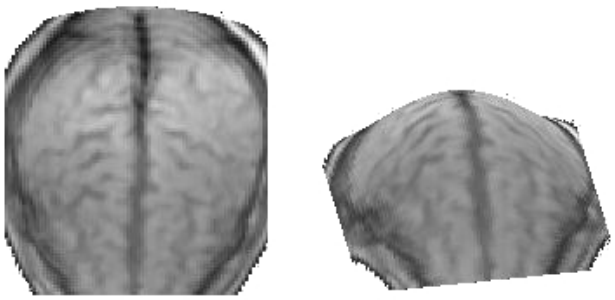

Figura 2: Corte em uma cabeça através de um plano curvilinear arbitrário.

Generalizando, a geração de ambientes virtuais para o auxílio em procedimentos cirúrgicos

e diagnósticos deve prover realismo e formas de interação (BARFIELD; HENDRIX, 1995). Satava (1993, 1994) identificou cinco elementos que afetam o realismo em um ambiente virtual para aplicações médicas, os quais são mostrados na tabela 2.

Fidelidade	(Alta resolução dos gráficos)
Exposição de propriedades dos órgãos	(Deformações de morfologia ou a cinemática das juntas)
Exposição de reações dos órgãos	(Vazamento de sangue de uma artéria ou de bile do fígado)
Interatividade entre objetos	(Tal como os instrumentos cirúrgicos e os órgãos)
Retro-propagação sensorial	(Tal como a utilização de *force-feedback*)

Tabela 2: Elementos que afetam o realismo em ambientes virtuais para aplicações médicas (SATAVA, 1993, 1994).

Mesmo que elementos como a exposição de propriedades e reações de órgãos, a interatividade entre objetos e a retro-propagação sensorial dependam de informações específicas dos objetos envolvidos, em termos práticos, todos dependem de estruturas de dados computacionais capazes de armazenar e indexar, com alta precisão, dados visuais e informações que satisfaçam os requisitos necessários para a obtenção dos elementos. Desconsiderando questões relacionadas à fidelidade, todos os elementos são dependentes da aplicação a que se destina a modelagem de um determinado ambiente virtual, no entanto, todos possuem uma base genérica.

Uma base genérica para a modelagem de ambientes virtuais com aplicações médicas inicia-se com a capacidade de identificar e representar objetos tridimensionais distintos dentro de um conjunto de imagens bidimensionais. Para cada um destes objetos deve haver a possibilidade de associação de informações, cujo cunho é apenas conhecido no domínio de uma determinada aplicação. Além disso, a visualização depende da construção de superfícies, as quais devem expor a qualidade de ser facilmente modificadas (apresentando operadores simples), possibilitando a modelagem de formas de interação entre o usuário e os objetos. Formas de interação provêem da capacidade de manipular as superfícies.

Robb e Cameron (1994, 1995) mencionam detalhes técnicos relacionados ao desempenho para obtenção de realismo em um ambiente virtual. Tais detalhes especificam que a taxa de

atualização de vídeo deve se manter em torno de trinta *frames* por segundo e o tempo de resposta a uma ação do usuário deve ser menor que cem milisegundos, conforme mostra a tabela 3.

Aspecto	Desempenho
Taxa de atualização de vídeo	$\sim$ 30 frames/s
Tempo de resposta à ação	$<$ 100 ms

Tabela 3: Desempenho requerido para obtenção de realismo em ambientes virtuais (ROBB; CAMERON, 1994, 1995).

Segundo Arthur (apud ROBB; CAMERON, 1994, 1995), Holloway, Fuchs e Robinett (apud ROBB; CAMERON, 1994, 1995) e Kaltenborn e Rienhoff (apud ROBB; CAMERON, 1994, 1995) tempos de resposta maiores que cento e vinte milisegundos fazem com que o usuário compense excessivamente o atraso a tempos de resposta maiores que trezentos milisegundos podem causar desconfortos e induzir enjôos. Além disto, Robb e Cameron identificam a quantidade de dados a ser renderizada como um dos fatores responsáveis pelo desempenho da aplicação. O *hardware* computacional pode ser decisivo para a obtenção de tais requisitos, porém, técnicas de computação gráfica em tempo real podem ser empregadas para reduzir esta dependência. Robb e Cameron (1994, 1995) ainda acrescentam que, para aumentar o grau de imersão e a ilusão de realidade, o visor deve ser acoplado à cabeça do usuário. Segundo Sollenberger e Milgran (apud ROBB; CAMERON, 1994, 1995) e Arthur, Booth e Ware (apud ROBB; CAMERON, 1994, 1995), tarefas como rastreamento, localização e movimentação têm taxas de erro melhoradas através do uso de visores acoplados.

A modelagem e implementação de uma base genérica para construção de ambientes virtuais com aplicações médicas possibilitam a expansão ágil desta para ambientes voltados à aplicações específicas. Esta expansão depende da inclusão de informações dependentes da aplicação e das formas de tratamento destas informações no momento da interação com os objetos virtuais. As formas de interação também são dependentes da aplicação.

Este trabalho mostra a modelagem de sólidos (objetos), tanto por interior como por bordas, e seus operadores como peça central para a obtenção de um ambiente virtual para realização de planejamento cirúrgico, dependendo apenas da interpretação dos dados primários, no caso, representados pelas seqüências de imagens médicas. Mäntylä (1988) utilizou uma abordagem sim-

ilar para o desenvolvimento do *Geometric WorkBench*. No entanto, a representação de sólidos deve ser flexível o suficiente para representar as mais diversas formas, provendo possibilidade de manipulação ágil. Para isto, é apresentada uma variação de uma estrutura para representação de sólidos baseada em faces, com operadores simples, para representar superfícies conhecidas como *non-manifold* (ver seção 2.2.2), além de permitir operações de conjuntos, as quais são essenciais para a modelagem avançada de sólidos. Também, para a montagem das estruturas de dados para representação de sólidos escolhidas, devem acompanhar um conjunto de métodos capazes de inserir dentro daquelas objetos arbitrários, identificando as peculiaridades destes e representando-os de forma a não perder suas propriedades, como a forma das faces por exemplo, o que ocorre em determinadas estruturas.

A modelagem de ambientes virtuais para simulação cirúrgica ainda impõe outros requisitos para a representação de sólidos, como a capacidade de aumento e redução de pontos em uma determinada superfície, o qual é necessário para a simulação do comportamento do tecido. Estes requisitos serão levantados e demonstrados através de um grafo de dependências que abrange desde a interpretação dos dados primários (imagens médicas) até a imersão realista. Também será demonstrado como a estrutura de dados para representação de sólidos previamente mencionada se adapta a estes requisitos. Ferramentas adicionais são adicionadas para garantir a obtenção da sensação de tempo real na aplicação.

1.1 Objetivos

Serão apresentados os objetivos deste trabalho, classificando-os entre objetivo geral e específicos.

1.1.1 Objetivo Geral

Demonstrar as dependências computacionais necessárias para a obtenção de um sistema capaz de gerar um ambiente cirúrgico imersivo, tendo como ponto de partida os dados médicos tridimensionais, contendo os elementos que influenciam o realismo neste tipo de aplicação,

definidos por Satava (1993, 1994) (tabela 2 na página 17), e atendendo aos requisitos de desempenho definidos por Robb e Cameron (1994, 1995) (tabela 3 na página 18). Além disto, ressaltar a importância da representação, visualização e manipulação de sólidos (objetos) dentro de sistemas para realidade virtual imersiva, propondo estruturas e métodos, independentes da aplicação médica.

1.1.2 Objetivos Específicos

Dentro da meta proposta, tais objetivos são visados:

- Levantar nas técnicas de computação gráfica e realidade virtual atualmente existentes, os modelos e métodos necessários para a obtenção dos elementos definidos por Satava (1993, 1994) (tabela 2 na página 17), mapeando-os diretamente;
- Gerar um grafo de dependências entre os modelos e métodos previamente levantados, abrangendo desde os dados médicos tridimensionais até a geração do ambiente virtual, destacando, neste contexto, a importância das estruturas para representação de sólidos e adicionando ferramentas computacionais que visam atender aos requisitos de desempenho definidos por Robb e Cameron (1994, 1995) (tabela 3 na página 18);
- Avaliar a adequação de estruturas para representação de sólidos, tanto por interior como por bordas, e seus operadores, dentro das dependências levantadas, propondo um modelo suficientemente capaz de atender aos requisitos de representação necessários;
- Apresentar métodos para construção das estruturas para representação de sólidos, independentes dos dados médicos tridimensionais, adequados ao modelo de representação proposto.

2 *Revisão Bibliográfica*

Neste capítulo serão abordados os modelos conhecidos para simulação cirúrgica e os modelos volumétricos utilizados na computação gráfica para representação dos sólidos, seguindo a seguinte estrutura:

- Revisão e análise dos modelos utilizados para realização de simulação cirúrgica, ressaltando as propriedades e etapas comuns a tais modelos (seção 2.1);
- Levantamento dos modelos volumétricos para representação de sólidos, abordando tipos e propriedades, e avaliando o enquadramento destes dentro da simulação cirúrgica (seção 2.2);
- Levantamento de métodos para obtenção de modelos volumétricos para representação de sólidos a partir de dados médicos tridimensionais, tais como as séries de tomografia computadorizada e ressonância magnética, abordando técnicas para adequação dos resultados gerados aos modelos mencionados na seção 2.2, simplificação de malhas e geração de malhas de controle para superfícies de subdivisão[1] (seção 2.3).

2.1 Modelos para Simulação Cirúrgica

Georgii e Westermann (2005) ressaltaram, de forma genérica, os seguintes passos para a inicialização de simulação e renderização de corpos heterogêneos deformáveis (resumido):

1. Construção de uma malha de elementos finitos [2];

[1] Ver seção 2.2.2.

[2] Um dos métodos largamente utilizados para simulação de comportamento de superfícies e tecidos quando

2. Atribuição das propriedades do corpo a ser simulado à malha de elementos finitos;

3. Construção de uma malha geométrica para visualização (seções 2.2 e 2.2.2);

4. Associação entre a malha de elementos finitos e a malha geométrica.

Estes passos indicam a utilização de dois componentes para que a simulação possa ser realizada: um modelo para simulação do sólido e um modelo para visualização do sólido, sendo estas independentes da origem das informações (propriedades físicas e morfológicas do sólido).

Em aplicações médicas, as informações relacionadas às propriedades morfológicas dos objetos (órgãos, tecidos, etc) provêem da análise de dados médicos tridimensionais (como por exemplo as seqüências de imagens de tomografia ou ressonância magnética), enquanto as informações relacionadas às propriedades físicas dependem de análises não relacionadas à ciência da computação. A análise da morfologia de um determinado objeto depende da interpretação dos dados médicos e da capacidade de distinguir, computacionalmente, os objetos presentes nos dados. Este processo de interpretação dos dados visando a identificação dos objetos presentes nestes, conhecido por segmentação, é o predecessor da etapa de construção de uma estrutura volumétrica para representação de forma (uma malha, por exemplo) (GIBSON et al., 1998; ZERFASS; KEEVE, 2001; AL-KHALIFAH; ROBERTS, 2004; ÇAKMAK; KÜHNAPFEL, 2000; AYACHE, 1997; NAKAJIMA et al., 1999), já que métodos utilizados para geração deste tipo de estrutura, tal como o *Marching Cubes* (LORENSEN; CLINE, 1987), não são capazes de fazer esta distinção entre os objetos (ver seção 2.3). Métodos de segmentação voltados a imagens médicas são levantados por Ayache et al. (apud AYACHE, 1997) e Lakare (2000).

Para a simulação de tecidos (simulação de sólido), a qual depende das propriedades físicas dos objetos, quatro modelos são amplamente utilizados: massa-mola (MILLER, 1988; CHADWICK; HAUMANN; PARENT, 1989; PROVOT, 1995; DESBRUN; SCHRÖDER; BARR, 1999), elementos finitos (ZIENKIEWICZ, 1977 apud TERZOPOULOS et al., 1987)(DHATT; TOUZOT, 1984 apud MCINERNEY; TERZOPOULOS, 1993)(ZIENKIEWICZ, 2000), elasticidade linear tridimensional (LANDAU; LIFSHITZ, 1986 apud AL-KHALIFAH; ROBERTS, 2004) e *chainmail* (GIBSON,

submetidos à ações de forças.

1997). Não serão abordados detalhes destes métodos, já que a escolha de um destes depende da aplicação. Detalhes sobre estes métodos, vantagens e desvantagens e aplicações conhecidas são descritas nos trabalhos publicados por Gibson e Mirtich (1997), Delingette (1998), Al-Khalifah e Roberts (2004) e Nealan et al. (2005).

Após a geração de uma estrutura para visualização (segmentação e análise morfológica) e aplicação de propriedades do sólido através de modelos para simulação de tecidos, o próximo passo para obtenção de um simulador cirúrgico é a construção de uma plataforma para visualização e interação com o usuário (interface com o usuário), a qual deve ser capaz de proporcionar a manipulação dos sólidos (AYACHE, 1997; GIBSON et al., 1998; YOO; RHEINGANS, 1999; NAKAJIMA et al., 1999; ÇAKMAK; KÜHNAPFEL, 2000; ZERFASS; KEEVE, 2001; AL-KHALIFAH; ROBERTS, 2004). O tipo de manipulação a ser suportada pela interface com o usuário depende da aplicação, sendo estas, por exemplo, a abertura de cortes e fixação de clips um uma aplicação para treinamento cirúrgico, as quais não precisam ser suportadas em aplicações para treinamento de endoscopia. Também a navegação pelo ambiente é dependente da aplicação, podendo ser mais flexível ou restrita. Apesar de dependentes da aplicação, todas as formas de interação são dependentes da detecção de colisões, de forma que se determine a ocorrência de contato entre o manipulador e o objeto a ser manipulado ou entre dois objetos (DELINGETTE, 1998). Métodos para detecção de colisão são amplamente utilizados em sistemas gráficos, principalmente em jogos, e podem ser encontrados nos trabalhos de Moore e Wilhelms (1988), Foley et al. (1990), Eberly (2000, 2004) e Lengyel (2001). Métodos para detecção de colisão aplicados à cirurgias virtuais são demonstrados por Lombardo, Cani e Neyret (1999) e aplicados a objetos deformáveis por Teschner et al. (2005).

Gibson et al. (1997, 1998) propuseram o modelo para simulação de ambientes cirúrgicos virtuais disposto na figura 3, o qual é iniciado através dos passos de segmentação de imagens e geração de modelos volumétricos. Após as etapas iniciais, o sistema entra em um ciclo fechado onde o usuário realiza modificações nos parâmetros visuais (navegação) e nos parâmetros do modelo volumétrico (manipulação dos objetos), as quais são refletidas sobre o modelo volumétrico, fazendo com que a visualização seja gerada a partir do modelo atualizado. Este

sistema é a base dos sistemas para geração de ambientes cirúrgicos virtuais.

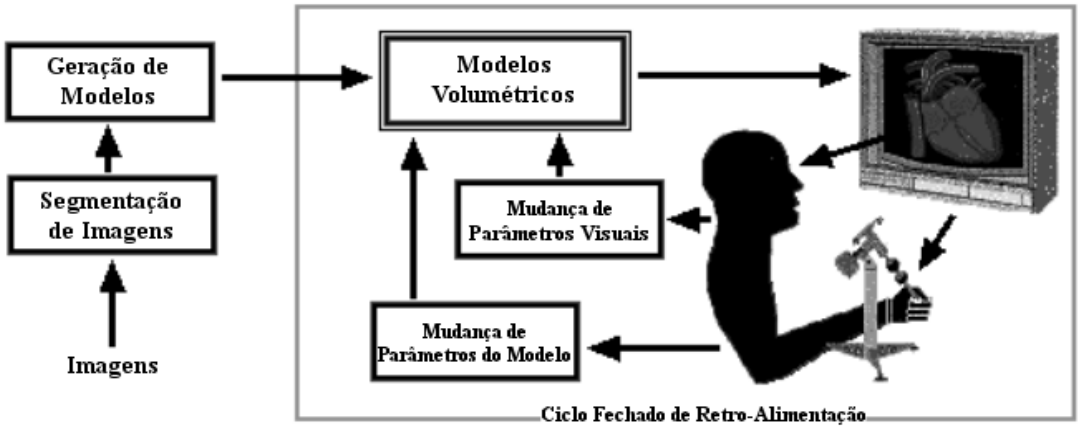

Figura 3: Modelo para sistema de simulação de ambientes cirúrgicos virtuais proposto por Gibson et al. (1997, 1998).

Zerfass e Keeve (2001) apresentam um modelo mais detalhado, porém com uma base idêntica, daquele apresentado por Gibson et al. (1997, 1998), o qual pode ser visto na figura 4. Neste modelo, além das etapas de segmentação, geração de modelo geométrico e do ciclo de interação e atualização do modelo (o qual aparece representado pelas etapas de geração de cena tátil, interação tátil e deformação), estão inclusas, também, a etapa de atribuição de propriedades do sólido (no caso, o modelo utilizado para esta tarefa é o de elementos finitos), formas de manipulação de sólido (em interação tátil) e as etapas que envolvem a manipulação do sólido (deformação): detecção de colisão, avaliação das condições das bordas (para determinar se a malha precisa ser refinada ou não), adaptação da malha e geração de informação tátil (retro-propagação de força ou *force-feedback*).

Yoo e Rheingans (1999) apresentaram um modelo onde, além dos dados extraídos das imagens, dados ao vivo são inseridos no sistema (figura 5), possibilitando a realização de cirurgia auxiliada por computador. A inserção de dados ao vivo se dá através da utilização de rastreadores, tanto no paciente como nos instrumentos. Modelo similar é proposto por Robb e Cameron (1994, 1995).

De acordo com os modelos e propostas apresentados, podemos generalizar as etapas realizadas por um simulador cirúrgico da seguinte forma:

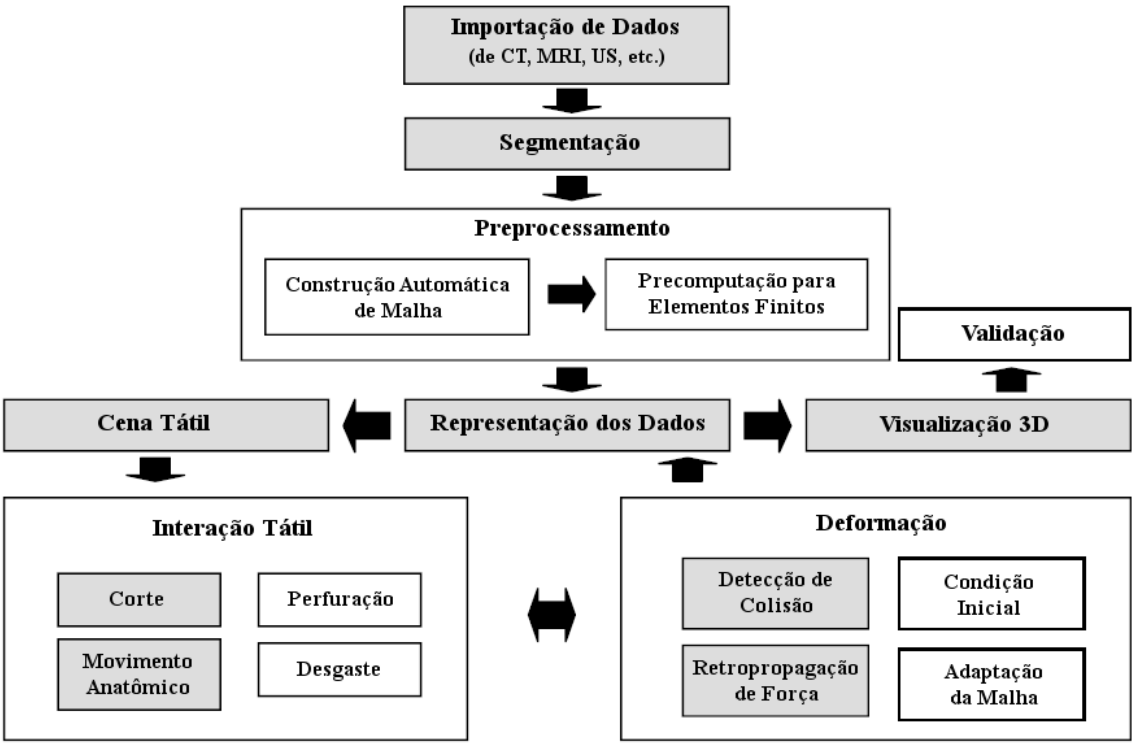

Figura 4: Modelo para sistema de simulação de ambientes cirúrgicos virtuais proposto por Zerfass e Keeve (2001).

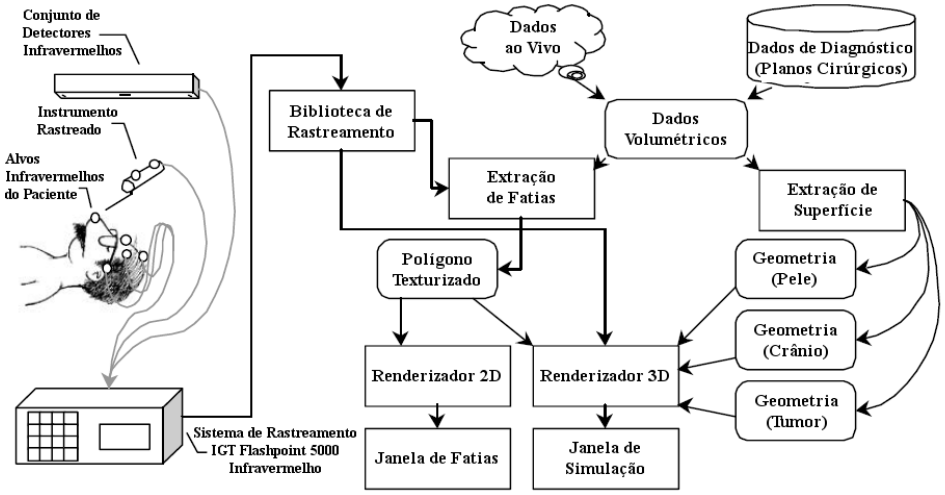

Figura 5: Modelo para sistema de simulação de ambientes cirúrgicos virtuais proposto por Yoo e Rheingans (1999).

1. Tendo como entrada dados médicos tridimensionais capazes de prover informação morfológica (como, por exemplo, as seqüências de imagens médicas de tomografia ou ressonância magnética), aplicar um processo de segmentação sobre estes, visando identificar objetos (sólidos) distintos;

2. Com o conjunto de entrada segmentado (objetos distintos identificados dentro dos dados de entrada iniciais), gerar modelos volumétricos para representação dos objetos;

3. Aplicar aos modelos volumétricos propriedades físicas através de modelos para simulação de sólidos (considerando que os dados referentes às propriedades físicas do sólido em questão estejam disponíveis);

4. Gerar a visualização dos sólidos;

5. Através de métodos de detecção de colisão, avaliar a interação entre o usuário (através de ferramentas dependentes da aplicação) e os sólidos, atualizando os modelos volumétricos e a visualização e gerando informações sensoriais (táteis, por exemplo) quando pertinente.

Os modelos volumétricos para representação de sólidos são abordados na seção 2.2.

2.2 Modelagem de Sólidos (Estruturas de Dados Espaciais)

Estruturas de dados espaciais são utilizadas para indexar dados referentes a localizações no espaço e relações espaciais entre estes dados, mesmo que algumas dessas relações não sejam atribuídas pelo usuário (SAMET, 1989a). Tais relações referem-se, por exemplo, à composição ou vizinhança entre objetos. Dados representados por tais estruturas consistem em objetos compostos por pontos, linhas, regiões, retângulos, superfícies, volumes e, inclusive, dados pertinentes a dimensões de maior ordem, tal como o tempo.

A representação de objetos presentes em imagens exige estruturas capazes de representar regiões. No caso de imagens médicas tridimensionais, é necessária a capacidade de representação de volumes. Segundo Samet (1989a), uma região (ou volume caso esteja-se utilizando

três dimensões) pode ser representada por seu interior ou suas bordas. Uma representação de bordas nada mais é que um polígono (em duas dimensões) ou um poliedro (em três dimensões), fornecendo informações apenas sobre a anatomia de um determinado objeto, o que não é suficiente para as aplicações médicas envolvendo simulação cirúrgica. Tais aplicações exigem o conhecimento da estrutura como um todo.

No entanto, caso os objetos a serem representados não sejam retilíneos, as representações por interior tornam-se apenas uma aproximação do objeto em suas bordas. Para que este efeito não ocorra neste tipo de representação seria necessária precisão infinita. Um exemplo pode ser visto na figura 6, onde representa-se uma curva através de uma quadtree. Para este tipo de situação, uma representação por bordas seria mais adequada, onde utiliza-se modelos geométricos para a composição do objeto. Wilhelms e Gelder (1992) mostram a utilização de ambos os tipos de estrutura para a representação de objetos extraídos de imagens.

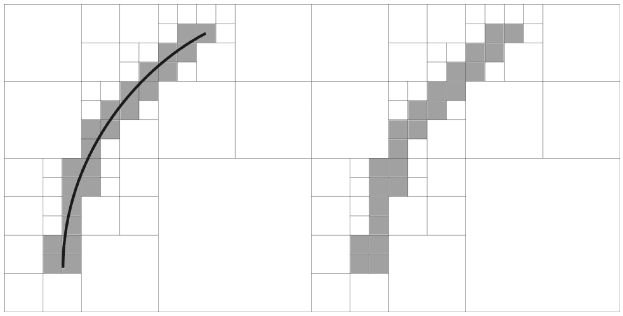

Figura 6: Representação de uma curva através de uma *quadtree*.

Requicha (1980) ainda demonstra um comparativo entre formas de representação. Representações de volumes por interior (no caso, *spatial enumeration* e *cell decomposition*) têm vantagem no cálculo de propriedades homológicas e de massa e na análise por elementos finitos, enquanto as representações por bordas são mais eficientes para renderização e interação. As outras representações mencionadas, como *primitive instancing* e CSG (*Constructive Solid Geometry*) não se adaptam à reconstrução de dados médicos pois se baseiam em modificação de sólidos primitivos até que seja atingida a forma desejada, o que se torna complexo no caso da modelagem de estruturas não triviais, como a massa encefálica ou o crânio. A representação *sweep* é utilizada para a modelagem de sólidos de revolução e modelagem através de deslocamentos, o que também não se adapta a modelagem anatômica, já que seria necessária uma

"fôrma" inicial para toda a estrutura.

Neste trabalho adota-se uma abordagem mista entre representação por interior e bordas, utilizando-as onde são melhor aplicáveis. Srihari (1981) fez um levantamento das estruturas para representação de imagens digitais tridimensionais, onde ressalta as vantagens da unificação entre representações por bordas e por interior e da capacidade de armazenar informações estruturais referentes aos objetos representados.

2.2.1 Representação de Volumes por Interior

Samet (1989a, 1990) cita a utilização de estruturas conhecidas como *octrees* para a representação de volumes, por ser compacta e possibilitar rápido acesso aos dados. Estruturas conhecidas como *BSP-trees* (*Binary Space-Partitioning trees*) (FUCHS; KEDEM; NAYLOR, 1980) podem ser estendidas para a representação de volumes, já que Thibault e Naylor (1987) demonstraram o uso destas para a representação de poliedros arbitrários.

Para a representação volumétrica, ambas estruturas requerem uma definição do objeto a ser representado. Tal definição dos objetos, no caso de utilização de imagens como fonte de dados, ocorrerá somente após o processo de segmentação das imagens. Caso não haja uma definição dos objetos, todos os dados encontrados nas imagens são considerados um único objeto. O processo de segmentação faz o agrupamento de *pixels* em regiões. No entanto, as *octrees* podem ser utilizadas para a segmentação das imagens de forma direta, sem o envolvimento de outros métodos, conforme demonstrado por Horowitz e Pavlidis (1976), que fazem utilização das *quadtrees*, estruturas similares às *octrees*, porém bidimensionais (FOLEY et al., 1990), para obter as regiões agrupadas. Posteriormente, Weingärtner e Dillmann (1995) demonstraram a utilização das *octrees* para uma generalização tridimensional do mesmo método de segmentação. Salembier e Garrido (2000) demonstram a utilização das *BSP-trees* no processo de segmentação, no entanto, uma prévia divisão das regiões se mantém necessária.

Além da possibilidade de unificação dos processos de segmentação e geração de representação volumétrica dos objetos oferecida pelas *octree*s, estas são potencialmente mais compactas

que as *BSP-trees* (FOLEY et al., 1990) e mais simples para a aplicação de operações de conjuntos (SAMET, 1989a, 1990).

2.2.1.1 *Octrees*

As *octrees* surgiram com o propósito de resolver problemas do método de representação conhecido como *Spatial-Occupancy Enumeration* (enumeração da ocupação espacial), tais como a consulta de dados e ocupação de recursos (FOLEY et al., 1990). Este esquema de representação foi apresentado por March e Steadman (1971) e consiste em dividir o espaço em um conjunto de cubos do mesmo tamanho. A informação é então armazenada em um vetor, cuja dimensão é igual a quantidade de subdivisões do espaço e cada elemento está associado a uma destas. Cada um dos elementos do vetor indica a presença ou não de um cubo na região associada. Quanto maior a quantidade de subdivisões, melhor será a precisão da representação, no entanto, maior será o vetor e o tempo de consulta. A figura 7 mostra um *donut* representado por *spatial-occupancy enumeration*.

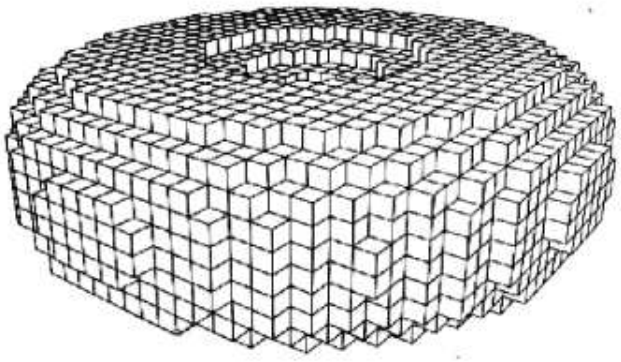

Figura 7: *Donut* representado por *spatial-occupancy enumeration* (CHRISTESSEN, 1980).

Resolvendo os problemas do *spatial-occupancy enumeration*, surgiram as *octrees* no final da década de 70, sendo estas derivadas das *quadtree*, conhecidas desde o final da década de 60 e utilizadas para representações bidimensionais (FOLEY et al., 1990). Segundo Samet (1984), as *octrees* foram desenvolvidas independentemente por diversos pesquisadores, tais como Hunter (1978), Reddy e Rubin (1978), Jackins e Tanimoto (1980) e Meagher (1980).

O método de representação por *octree* consiste em subdividir recursivamente o espaço em

oito partes até que cada uma das subdivisões satisfaça um conjunto de primitivas previamente determinadas (SAMET, 1989a). A quantidade de recursões pode ser limitada pela precisão do *hardware* de execução. O resultado é uma representação hierárquica do espaço. O processo de construção de uma representação por *octree* é mostrado na figura 8, onde a raiz da árvore representa o espaço inicial a ser representado e cada um dos descendentes um subespaço obtido através da subdivisão do espaço anterior. Caso um determinado subespaço não satisfaça o conjunto de primitivas estabelecido, este é subdividido. Uma representação por *octree* requer que o espaço inicial tenha o formato de um cubo ou um paralelepípedo.

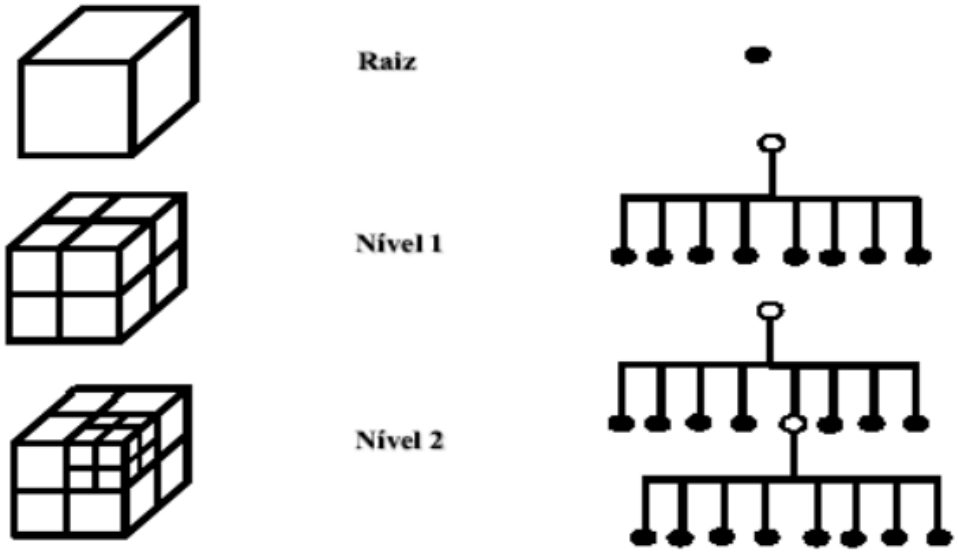

Figura 8: Representação do espaço por uma *octree* (WAGNER, 2001).

Além das vantagens obtidas em relação ao *spatial-occupancy enumeration*, as *octrees* permitem a realização de um conjunto de operações, tais como as transformações booleanas (SAMET, 1984, 1989a, 1990) e a busca de vizinhança entre nodos e travessias pela árvore (SAMET, 1989b; BHATTACHARYA, 2001; FRISKEN; PERRY, 2002).

Gargantini (1982) introduziu as *octrees* lineares, as quais se baseiam em armazenar um código para cada nodo, que serve para localizar este nodo dentro da estrutura. Esta representação através de código desfaz a necessidade do uso de ponteiros e do armazenamento dos nodos intermediários, alocando apenas os nodos folha, reduzindo a capacidade total de armazenamento necessária. O restante da estrutura pode ser inferida através do código armazenado em

cada folha. No entanto, a utilização do método proposto por Gargantini assume que as dimensões do espaço representado sejam iguais e sejam uma potência de dois (2^n x 2^n x 2^n) para que a geração do código seja correta, o que nem sempre pode-se afirmar em um conjunto de imagens, além de que a travessia na árvore exige um número maior de operações, já que a estrutura deve ser inferida.

Wilhelms e Gelder (1992) apresentaram uma *octree* mais adequada a representação de objetos em seqüências de imagens, conhecida como BONO (*Branch-On-Need Octree*). A BONO faz a divisão do espaço de forma que alguns dos nodos cubram o maior volume em potência de dois possível, enquanto os nodos opostos cobrem o espaço restante. Segundo a técnica proposta por Wilhelms e Gelder, as dimensões de cada nodo podem ser encontradas através da análise do alcance das dimensões do nodo pai e de sua posição dentro do espaço definido pelo pai. Portanto, para cada nodo, basta seu alcance (nos eixos X, Y e Z), sua origem (um ponto de referência que marca o início do nodo) e o seu código de posição dentro do pai.

Sendo o código de posicionamento uma tripla binária ($[z, y, x]$), o código 001 refere-se ao nodo filho que se encontra na posição inferior no eixo Z, inferior no eixo Y e superior no eixo X. Supondo as dimensões do espaço inicial como 320 x 320 x 40, determina-se as dimensões do filho na posição 001 através da análise das dimensões iniciais em binário (tabela 4), sendo que as direções que se dividem são aquelas que tem o valor 1 no *bit* mais significativo do alcance. O alcance do filho do filho é dado pela a eliminação do *bit* mais significativo do alcance do pai, ressaltando que, quando o *bit* mais significativo for 1 e o código de posicionamento do filho, para aquela direção, for zero, o valor do alcance torna-se uma cadeia de 1s, de tamanho uma unidade menor do que o código do pai. Os alcances para o nodo filho cujo código de posicionamento é 001 é mostrado na tabela 5.

Nota-se que o método de geração da BONO não gera uma árvore completa, pois ao se analisar os alcances das dimensões do espaço referido pela tabela 4, apenas há divisão nas direções x e y, resultando em apenas quatro nodos filhos para o nodo que representa este espaço.

Wilhelms e Gelder (1992) ainda demonstram como fazer um armazenamento eficiente da

Direção	Alcance em binário	Alcance em decimal
x	100111111	319
y	100111111	319
z	000100111	319

Tabela 4: Análise das dimensões em binário de um determinado espaço para cálculo das dimensões de um nodo filho dentro de uma BONO.

Direção	Alcance em binário	Alcance em decimal
x	00111111	63
y	11111111	255
z	00100111	39

Tabela 5: Alcance das dimensões do nodo cujo posicionamento é 001 e os alcances do pai são dados na tabela 4.

estrutura, tanto para espaço como para travessia, e como otimizar o processo de geração de isosuperfícies através do uso da BONO. A figura 9 mostra a representação de uma artéria através de uma BONO e da isosuperfície resultante pelo método proposto.

Yau e Srihari (1983) também apresentaram uma *octree* adequada a representação de objetos em imagens multidimensionais, conhecida como *hyperoctree*. A utilização da *hyperoctree* resulta em uma representação mais compacta dos objetos, no entanto, a travessia e modificação da árvore não pode ser feita de forma intuitiva, já que devem ser realizadas em uma determinada ordem, o que descarta a possibilidade de utilização de métodos clássicos para o manuseio de *octrees*.

2.2.2 Representação de Volumes por Bordas (*B-Reps*)

As estruturas de dados para representações de volume por bordas representam as associações entre faces, vértices e arestas. Mäntylä (1988) classificou estas estruturas em três tipos: baseada em polígonos (também conhecida como estruturas baseadas em faces), baseadas em arestas e baseadas em vértices. Esta classificação depende do elemento colocado como chave

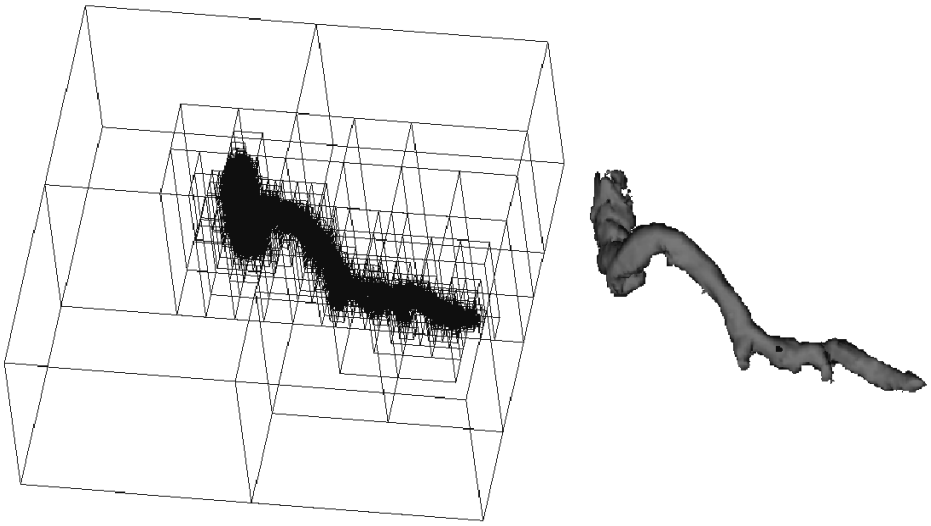

Figura 9: Representação de uma artéria através de uma BONO e da isosuperfície resultante.

central da estrutura. Por exemplo, se a partir de uma determinada face, deve-se percorrer as arestas para que se encontre seus vértices, ou a partir de um vértice, deve-se percorrer as arestas adjacentes para identificar a quais faces pertence este vértice, a estrutura em questão é classificada como baseada em arestas.

Cada tipo de estrutura tem sua eficiência dependente da aplicação em que é usada. No caso do planejamento cirúrgico, onde interage-se diretamente com as superfícies, modificando a topologia e a geometria das faces e reposicionando os vértices e arestas que pertencem diretamente às faces onde foi detectada a interação, se faz mais adequada uma estrutura baseada em faces. Higashi et al. (1995), além de apresentar uma estrutura de dados baseada em faces utilizando tabelas, ainda ressaltam outras vantagens da utilização desta classe de estruturas.

Segundo Mäntylä (1988), as *b-reps* representam as superfícies de um sólido através da divisão desta em uma coleção de faces (malha). Para manter a consistência das relações entre os componentes da estrutura (faces, vértices e arestas), as *b-reps* adotam um critério topológico, o qual é mantido através de operadores providos pela estrutura. Este critério topológico é, geralmente, que a malha sempre mantenha a característica de ser *2-manifold* (*two dimensional manifold*). Malhas com esta característica são utilizadas devido ao fato de que algumas operações de modelagem, como os métodos populares de subdivisão (ZORIN; SCHRÖDER, 2000),

requerem que as superfícies tenham esta propriedade e a utilização de malhas *non-manifold* (*non-2-manifold*, malhas que não apresentam a característica *2-manifold*) complicam os métodos de modelagem (HOFFMANN; VANECEK, 1990 apud AKLEMAN; CHEN, 2003b) (MÄNTYLÄ, 1986 apud AKLEMAN; CHEN, 2003b).

Outra definição bastante comum é que uma estrutura com propriedade *2-manifold* descreve um espaço semelhante ao euclidiano quando vista de perto, no entanto é uma estrutura complexa quando se analisa o todo. A utilização de *2-manifolds* é grande na matemática e na física por permitirem que estruturas grandes e complexas sejam modeladas e analisadas através de pequenas, e bem compreendidas, partes (LEE, 2000).

Malhas *2-manifold* são aquelas onde cada um dos vértices vizinhos a um determinado vértice formem uma topologia homeomorfa a um disco aberto (MÄNTYLÄ, 1988). A figura 10 ilustra este conceito, onde f_1, f_2 e f_3 são faces de um cubo e v_1, v_2 e v_3 são vértices da vizinhança de v_0. Pode-se perceber que a organização topológica da vizinhança de v_0 é homeomorfa a um disco aberto, logo este vértice apresenta característica *2-manifold*. Malhas com característica *2-manifold* apresentam as seguintes propriedades topológicas (MÄNTYLÄ, 1988):

- Cada aresta da estrutura se identifica topologicamente com exatamente uma outra aresta;
- As faces identificadas a partir de cada vértice podem ser organizadas em uma forma cíclica onde cada par consecutivo de faces no ciclo identifica uma aresta adjacente ao vértice;
- As faces da estrutura podem ser orientadas de forma que para cada par de aresta topologicamente identificada, uma delas está em orientação positiva na direção da face a qual pertence, e outra está em orientação negativa.

Além destas propriedades, as malhas com característica *2-manifold* sempre mantém a característica de Euler, a qual é expressa pela seguinte equação:

$$v - e + f = 2$$

onde v, e e f representam o número de vértices, arestas e faces, respectivamente. A característica de Euler pode ser, também, representada pela equação de Euler-Poincaré:

$$v - e + f = 2(s - h)$$

onde s representa o número de superfícies conectadas e h representa o *genus* (número de buracos) da superfície.

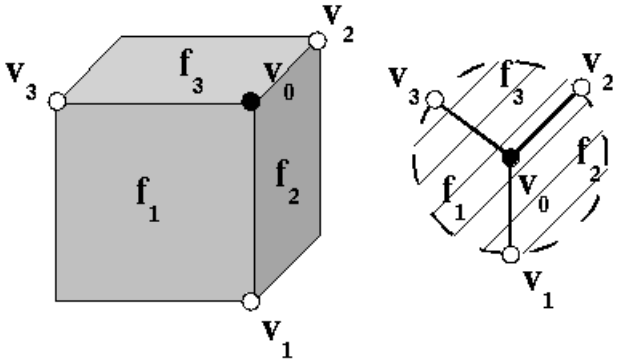

Figura 10: Vista geométrica e topológica de um vértice cujo sólido a qual pertence possuí propriedade *2-manifold*.

Baumgart (1972, (BAUMGART, 1974 apud MÄNTYLÄ, 1988), (BAUMGART, 1975 apud MÄNTYLÄ, 1988)) desenvolveu uma estrutura baseada em arestas, conhecida como *winged-edge*, juntamente com operadores para modificá-la, os quais foram denominados operadores de Euler, que mantém sempre a propriedade *2-manifold* dos sólidos por ela representados. Utilizando uma estrutura similar a da *winged-edge* e os operadores de Euler, Mäntylä (1988) desenvolveu uma estrutura capaz de representar *2-manifolds* com bordas. Esta estrutura, chamada de *half-edge*, apresenta as mesmas características das superfícies *2-manifold*, exceto nas bordas, onde as arestas não possuem uma idêntica topológica. Para a representação de *2-manifolds* com bordas, a fórmula de Euler-Poincaré é dada pela seguinte equação:

$$v - e + f = 2(s - h) - b$$

onde b representa o número de componentes nas bordas.

A *half-edge* introduziu um elemento chamado anel, o qual representa uma borda, tanto de

uma face quanto dentro de uma face. Quando dentro de um anel não existe uma face, este é interpretado como um buraco nesta. Com a introdução dos anéis, a fórmula de Euler-Poincaré é representada por:

$$v - e + f = 2(s - h) + r$$

onde r representa o número de anéis. Juntamente com a introdução dos anéis, Mäntylä modificou os operadores de Euler propostos por Baumgart, os quais eram muitos e pouco intuitivos, e adaptou-os à *half-edge*, gerando apenas dez operadores, sendo, de fato, cinco operadores e cinco operações inversas para cada um respectivamente. Sendo a estrutura da *half-edge* dada conforme a figura 11, os operadores definidos para esta estrutura são (com os inversos em parênteses):

- MVFS (KVFS): Cria um vértice, uma face e um sólido (inversamente, destrói um vértice uma face e um sólido);
- MEV (KEV): Cria uma aresta e um vértice (inversamente, destrói uma aresta e um vértice);
- MEF (KEF): Cria uma aresta e uma face (inversamente, destrói uma aresta e uma face);
- KEMR (MEKR): Destrói uma aresta e cria um anel (inversamente, cria uma aresta e destrói um anel);
- KFMRH (MFKRH): Destrói uma face e cria um anel e um buraco (inversamente, cria uma face e destrói um anel e um buraco).

A figura 12 demonstra os resultados topológicos gerados através da aplicação destes operadores.

Baseadas na *winged-edge* de Baumgart, outras estruturas de dados para representação de malhas com propriedade *2-manifold* foram propostas, tais como a *quad-edge* de Guibas e Stolfi (1985) e a estrutura baseada em arestas de Weiler (apud AKLEMAN; CHEN, 2003b). No entanto, todas estas estruturas derivadas da *winged-edge* são baseadas em arestas. A dificuldade da

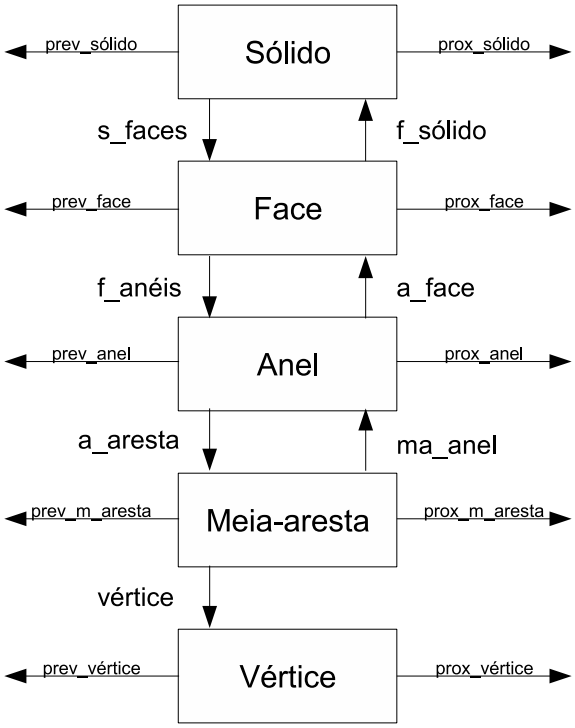

Figura 11: Visão hierárquica da estrutura de dados *half-edge*.

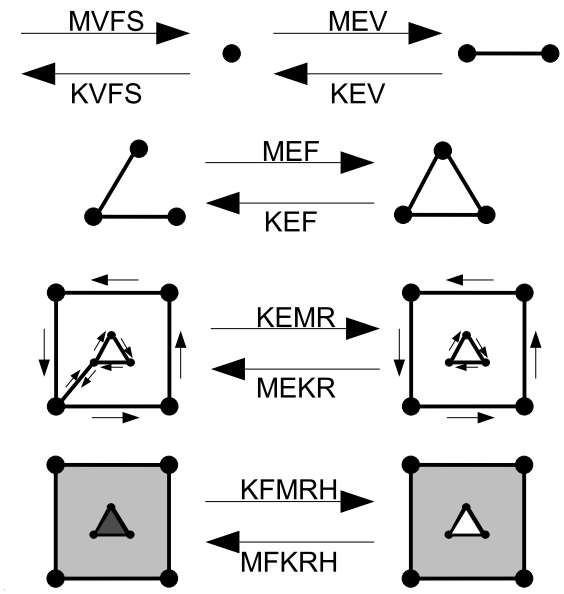

Figura 12: Resultados topológicos dos operadores de Euler.

implementação e utilização de uma estrutura baseada em faces se dá pelo fato de que, nos modelos propostos, a malha a ser representada já deve vir de forma consistente (*2-manifold*) e a utilização dos operadores propostos pode vir a gerar inconsistências (AKLEMAN; CHEN, 2003b). Akleman e Chen (1999) desenvolveram uma estrutura baseada em faces, denominada *Doubly Linked Face List* (DLFL), a qual garante sempre a propriedade *2-manifold* das malhas representadas, e ainda apresentam operadores simples e intuitivos para sua manipulação, sem que esta propriedade seja perdida. A DLFL é vista em detalhes na seção 2.2.2.1.

A modelagem de objetos *non-manifold* utilizando apenas topologia *2-manifold* pode se tornar complicada e gerar geometrias inadequadas (ver capítulo 4), apesar de que mesmo superfícies geometricamente *non-manifold* podem ser representadas por estruturas de dados que mantém topologia *2-manifold* (figura 13). Luo e Lukács (1991), Gueorguieva e Marcheix (1994), Yamagushi e Kimura (1995) e Floriani e Hui (2003) apresentaram estruturas e operadores para a representação de sólidos *non-manifold*, no entanto, a modelagem desta maneira não é intuitiva. Ying e Zorin (2001) apresentam um método para subdivisão de superfícies *non-manifold*, realizando subdivisão normal nas partes *2-manifold* e tratando separadamente os partes onde não ocorre esta característica.

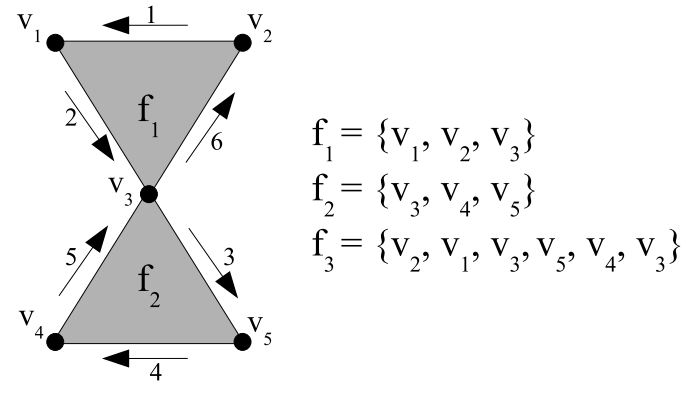

Figura 13: Representação de uma geometria *non-manifold* com topologia *2-manifold*.

Mäntylä (1988) propôs que a modelagem de objetos *non-manifold* seja realizada através da combinação entre objetos *2-manifolds*. Por exemplo, o objeto mostrado na figura 14, o qual não apresenta característica *2-manifold*, pois a aresta em destaque é compartilhada por quatro faces, pode ser modelado através de dois cubos (objetos *2-manifold*) e ter a conexão entre eles

armazenada separadamente. Higashi et al. (1993) e Lee e Lee (2001) demonstram este tipo de estrutura utilizando operadores de Euler clássicos (MÄNTYLÄ, 1988), para a modelagem das partes *2-manifold*, e operadores estendidos para as partes que não apresentam esta característica. Os operadores estendidos servem para adicionar um novo ciclo de arestas a um determinado vértice ou uma conexão entre arestas já pertencentes a duas faces (figura 14), entre outras operações. Higashi et al. (1993) ainda reformulam a equação de Euler-Poincaré para tratar este tipo de estrutura. A equação reformulada leva em conta a quantidade de elementos que faltam em uma determinada parte degenerada (*non-manifold*). Por exemplo, a aresta degenerada na figura 14, equivale na equação original a duas arestas. A equação de Euler-Poincaré reformulada é dada por:

$$(v_b - v_r) - (e_b - e_r) + (f_b - f_r) = 2(s - h)$$

onde v_b, e_b e f_b representam os componentes *2-manifold* e v_r, e_r e f_r representam os componentes degenerados.

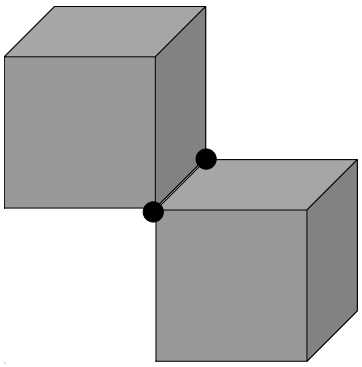

Figura 14: Sólido *non-manifold* composto por dois objetos *2-manifold*.

Conforme mencionado anteriormente, a propriedade *2-manifold* de uma determinada malha é útil para aplicação de técnicas populares de subdivisão. Segundo Zorin e Schröder (2000), as técnicas de subdivisão definem uma superfície suave através do limite de uma seqüência de refinamentos sucessivos (figura 15). Nos métodos mais populares, o limite da seqüência de refinamentos é uma superfície *b-spline*. Catmull e Clark (1978) e Doo e Sabin (1978) propuseram os primeiros métodos de subdivisão, ambos aplicados a malhas compostas por faces de geometrias variadas, resultando em uma superfície composta por faces quadrangulares. Loop (1987) propôs um método mais adequado para a subdivisão de superfícies compostas por faces

triangulares. Stam e Loop (2003) unificaram os métodos mencionados anteriormente para uma subdivisão adequada, tanto de faces arbitrárias como triangulares, e, posteriormente, Warren e Schaefer (2004) modificou este método para um melhor tratamento dos vértices e arestas nas bordas da superfície.

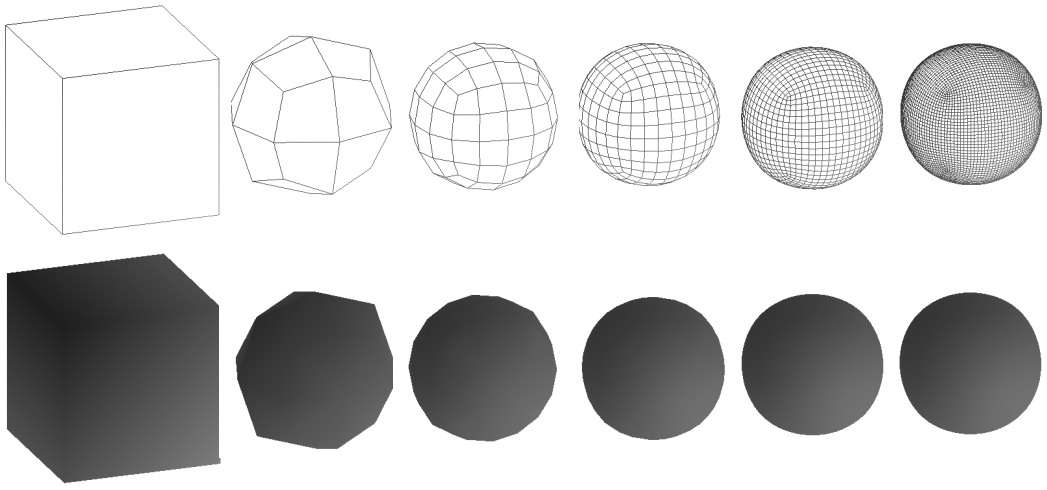

Figura 15: Subdivisão de um cubo através do método proposto por Catmull e Clark (1978).

Técnicas de subdivisão aplicadas a superfícies proporcionam as seguintes vantagens (ZORIN; SCHRÖDER, 2000):

- Generalização de superfícies *spline* para topologias arbitrárias;
- Nível de detalhes variados, dependendo da quantidade de refinamentos sucessivos;
- Uniformidade de representação, onde as superfícies podem ser tratadas tanto como superfícies bicúbicas ou um conjunto de polígonos;
- Estabilidade numérica necessária para a modelagem de elementos finitos, utilizados para a simulação de superfícies de sólidos;
- Implementação simples e computabilidade eficiente.

Além destas, a utilização de superfícies de subdivisão requer menor capacidade de armazenamento, sendo necessário apenas o armazenamento da superfície inicial enquanto os detalhes são inseridos após a seqüência de refinamentos.

Estruturas para representação de bordas adequadas para a aplicação de subdivisão devem, além de garantir propriedade *2-manifold*, prover rápido acesso a dados como valência de vértices e faces e vizinhanças. Este é o caso da *winged-edge*, *half-edge*, DLFL e outras.

2.2.2.1 *Doubly Linked Face List* (DLFL)

A DLFL, idealizada por Chen (1997) e posteriormente introduzida por Akleman e Chen (1999, 2000), é a implementação de um sistema de rotação de grafo (*graph rotation system*), o qual foi provado por Edmonds (apud AKLEMAN; CHEN, 1999) equivaler sempre a uma malha *2-manifold* orientada.

Uma rotação em um vértice pertencente a um determinado grafo é uma permutação cíclica das arestas incidentes a este vértice. Um sistema de rotação de um determinado grafo é uma lista de rotações, uma para cada vértice daquele (HEFFTER, 1891 apud AKLEMAN; CHEN, 1999).

As seguintes vantagens são obtidas através do uso da DLFL:

- Manipulação da estrutura através de quatro operadores;
- Computação eficiente de modificações topológicas (CHEN, 1997; AKLEMAN; CHEN, 1999, 2000);
- Ocupação de memória reduzida (CHEN, 1997);
- Operações como subdivisão e a inserção e remoção de buracos e "pegadores" (asa de uma xícara, por exemplo) são realizadas de forma eficiente (AKLEMAN; CHEN, 1999, 2000).

Além destas, a DLFL é uma estrutura baseada em faces e, conforme mencionado previamente, este tipo de estrutura é mais adequado para a aplicações de simulação cirúrgica. No entanto, a DLFL é eficiente apenas para representação de sólidos com topologia *2-manifold*, não sendo capaz de representar topologia *non-manifold*, e ineficiente para representação de topologias *2-manifold* com bordas (mais detalhes no capítulo 4).

Esta estrutura é basicamente composta por uma lista de faces, uma lista de vértices e uma lista de arestas (figura 16). Cada face da lista de faces é uma lista circular duplamente encadeada

de vértices, cada um representando o extremo de uma aresta, ou seja, cada par consecutivo de vértices representa uma determinada aresta. Cada vértice na lista de vértices é uma lista apontando para os respectivos extremos de arestas nas faces. Cada aresta na lista de arestas é composto por dois ponteiros bidirecionais apontando para os dois extremos de aresta que compõe esta, sendo que cada extremo apontado é o inicio desta aresta para cada um dos dois sentidos possíveis (e obrigatório no caso de representação de *2-manifolds*, conforme seção 2.2.2). A organização da estrutura desta forma permite que propriedades topológicas e geométricas, como adjacência entre faces ou valência de faces e vértices, sejam facilmente encontradas.

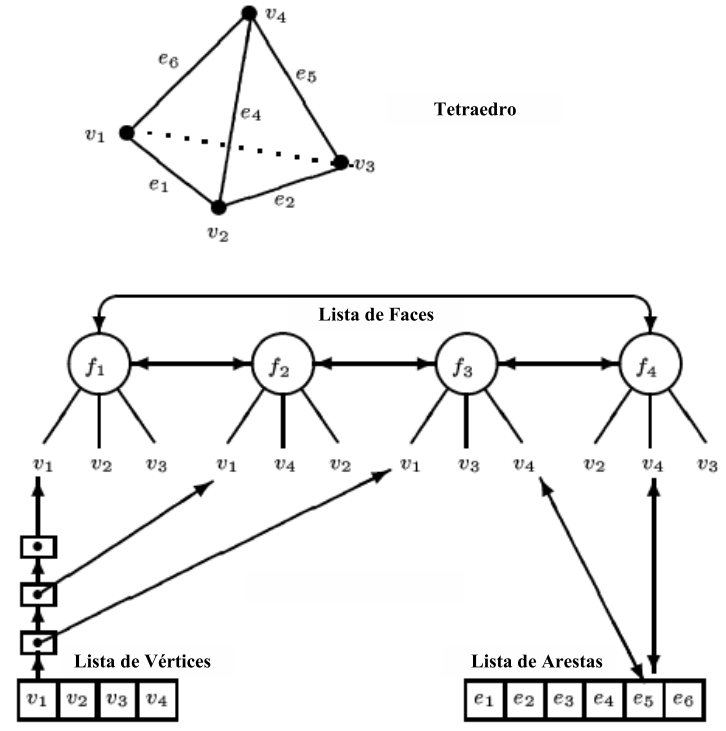

Figura 16: Estrutura de uma DLFL representando um tetraedro (AKLEMAN; CHEN, 2003b).

Os operadores para manipulação da DLFL são:

- CreateVertex(v): Cria uma nova face na lista de faces, composta unicamente pelo vértice v, e uma nova entrada na lista de vértices;
- DeleteVertex(v): Destrói um determinado vértice, a face que o possuí e a entrada na lista de vértices. Esta operação só pode ser aplicada no caso em que o nodo na lista de vértices possuí apenas um ponteiro e face possuí unicamente o vértice a ser destruído;

- InsertEdge(cor(e_1, e_2),cor(e_3, e_4)): Cria uma aresta entre o vértice (extremidade de arestas) que une as arestas e_1 e e_2 e o vértice que une as arestas e_3 e e_4, inserindo-a ma lista de arestas, de forma que, caso estas extremidades pertençam a mesma face, esta é dividida em duas, e caso pertençam a faces distintas, estas são unidas em uma única face;

- DeleteEdge(e): Destrói uma aresta, removendo-a da lista de arestas, de forma que, todos os seus extremos pertençam a mesma face, esta face é dividida em duas, e caso estes pertençam a faces distintas, estas são unificadas em uma única face.

Conforme demonstrado por Akleman, Chen e Srinivasan (2003), estes operadores formam um conjunto suficiente para a realização de todas as operações morfológicas em malhas com a propriedade *2-manifold*. Operadores de alto nível, os quais são baseados em chamadas a estes quatro operadores indicados (operadores de baixo nível), são introduzidos por Srinivasan (2004).

A figura 17 demonstra a inserção de uma aresta entre extremos pertencentes a uma mesma face, ocasionando a divisão desta, e a remoção de uma aresta pertencente a faces distintas, fazendo com que ambas as faces formem uma única. Pode-se observar que a rotação das faces permanece correta e a inserção de uma aresta faz com que esta se apresente nos dois sentidos possíveis (conforme requer a topologia *2-manifold*): de v_2 para v_5 na primeira face (f_1) e de v_5 para v_2 na outra (f_2).

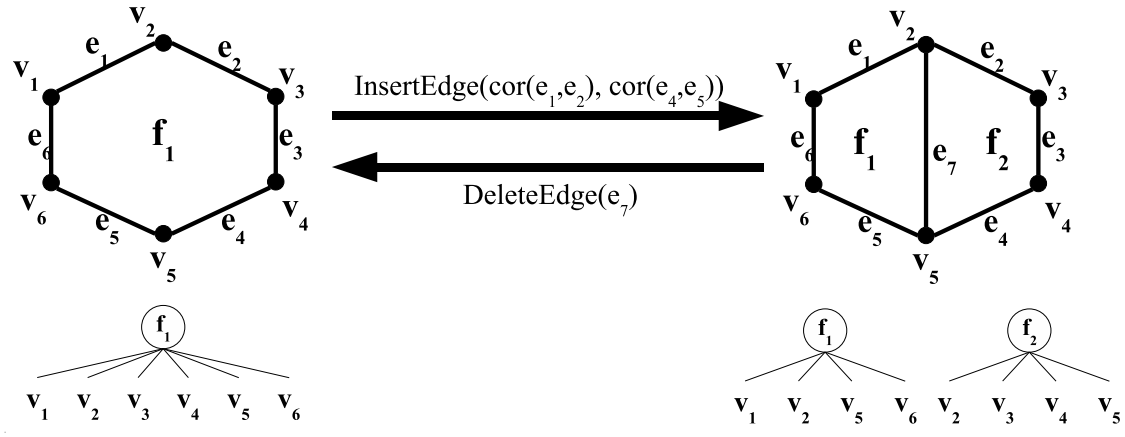

Figura 17: Inserção de aresta entre extremidades pertencentes à mesma face e remoção de aresta pertencente à faces distintas.

No caso de uma inserção de aresta entre extremos de faces distintas (figura 18), a unifi-

cação das faces pode gerar artefatos inadequados geometricamente, como uma face não planar, por exemplo, no entanto topologicamente mantendo a característica *2-manifold* e a correta orientação dos vértices (ciclos). Mesmo neste caso, a inserção de uma aresta faz a com que sua orientação ocorra nos dois sentidos, no entanto, neste caso ambos os sentidos ocorrem em uma mesma face: tanto no sentido de v_3 para v_6 como de v_6 para v_3 se encontram na mesma face (f_1). Pode-se perceber também que ao começar a percorrer a face f_1 a partir de qualquer vértice, o vértice inicial é atingido novamente, mantendo a propriedade cíclica das faces. Cada face em uma DLFL representa um anel e uma face da *half-edge*.

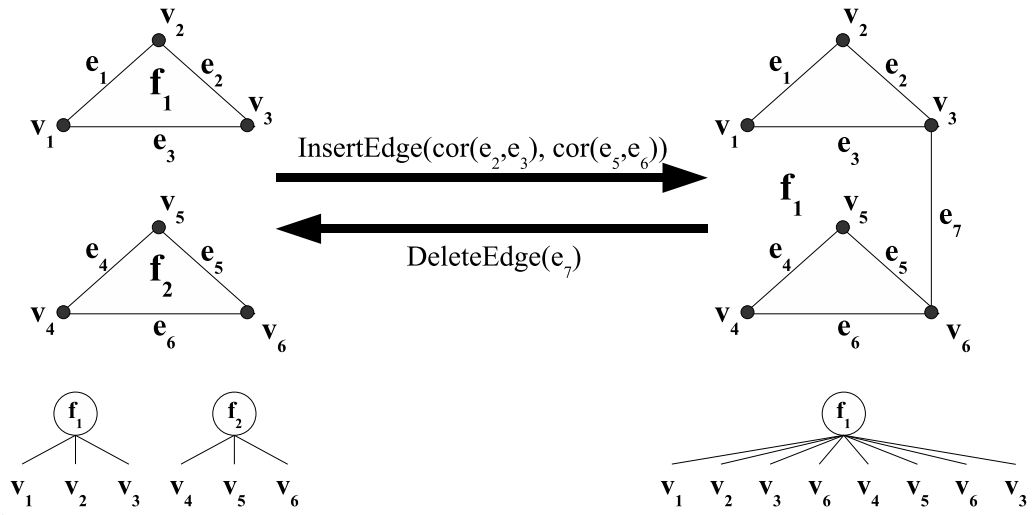

Figura 18: Inserção de aresta entre extremidades pertencentes à faces distintas e remoção de aresta pertencente à mesma face.

A eliminação de artefatos indesejados, como faces não planares, após consecutivas inserções e remoções de arestas se dá através de um método de subdivisão com eliminação de vértices (AKLEMAN; CHEN; SRINIVASAN, 2000; AKLEMAN et al., 2001, 2001; SRINIVASAN; AKLEMAN; CHEN, 2002). Também a inserção de buracos e "pegadores" na estrutura ocorre através do método de subdivisão, já que buracos não podes ser determinados através da interpretação da estrutura devido a ausência da entidade "anel", a qual existe na *half-edge*. Este processo pode ser visto na figura 19, onde a inserção de aresta da figura 18, se transforma em um "pegador" oco após a aplicação do método de subdivisão.

Akleman, Chen e Srinivasan ainda apresentam métodos para geração de cascas (AKLEMAN; CHEN; SRINIVASAN, 2003), refinamento de malha (AKLEMAN; CHEN, 2003a) e remodelagem de

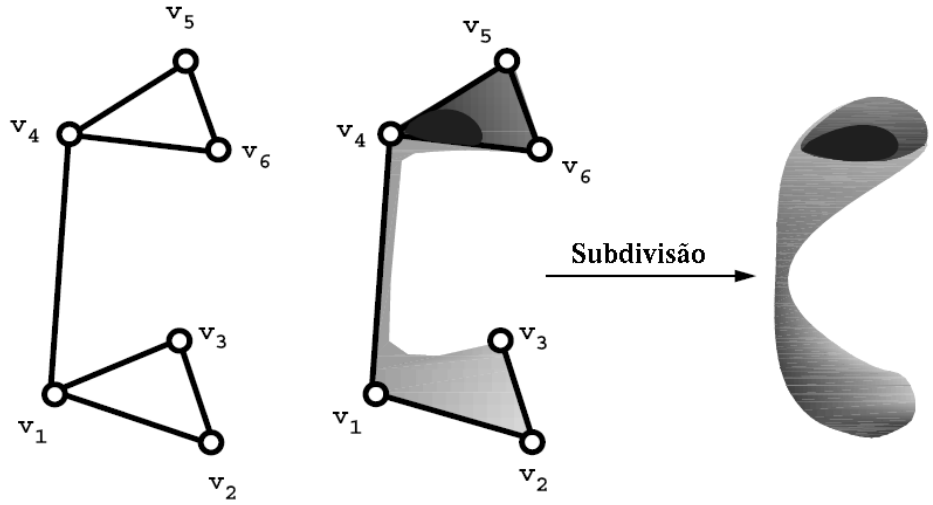

Figura 19: Subdivisão em uma estrutura não planar se transforma em "pegador" após processo de subdivisão (AKLEMAN et al., 2001).

malha (AKLEMAN; SRINIVASAN; MANDAL, 2005) baseados em DLFL.

2.3 Geração de Sólidos a partir de Dados Médicos Tridimensionais

Nesta seção serão levantadas as técnicas para geração de sólidos a partir de dados médicos tridimensionais segmentados. Juntamente, serão apresentados métodos para adaptar os dados gerados com as técnicas para geração de sólidos em estruturas de dados para representação de sólidos por bordas. A geração de estruturas de dados para representação de sólidos por interior é abordada na seção 2.2.1. Ainda é demonstrado o procedimento de adaptação de um sólido comum a uma malha de controle para superfície de subdivisão, a qual permite a construção do sólido com níveis varáveis de detalhamento (ver seção 2.2.2).

O método mais utilizado para geração de sólidos a partir de dados médicos tridimensionais é o *Marching Cubes* (LORENSEN; CLINE, 1987). Este método foi criado para a construção de isosuperfícies. Seu funcionamento se dá da seguinte forma:

1. Tendo como entrada um isovalor, percorre-se o conjunto de dados com um cubo de tamanho pré-estabelecido (quanto menor o cubo maior o tempo de processamento e a precisão);

2. À cada movimentação do cubo verifica-se o valor que se encontra no conjunto de dados tridimensionais na posição indicada por cada vértice daquele;

3. Aqueles vértices que possuem valor maior ou igual ao isovalor são ditos ativados, e considera-se que estes estejam dentro do sólido que se deseja construir. Os vértices com valor menor ao isovalor são ditos desativados e considerados fora do sólido;

4. Tendo um vetor com dimensão de oito *bits* (um para cada vértice do cubo), ativa-se os *bits* correspondentes aos vértices ditos ativados;

5. Este vetor é utilizado como índice de uma tabela com 256 posições que indica como as faces devem ser construídas para aquele caso de ativação;

6. Interpola-se linearmente sobre as arestas cujas extremidades possuem um vértice ativado e um desativado, visando encontrar a localização do isovalor.

A figura 20 ilustra a execução do *marching cubes* (em forma bidimensional, o qual é conhecido como *marching square*). Os vértices brancos estão desativados enquanto os pretos estão ativados. Observa-se em preto o polígono se aproximando da estrutura original. A aproximação deste polígono será maior quanto menores forem os cubos.

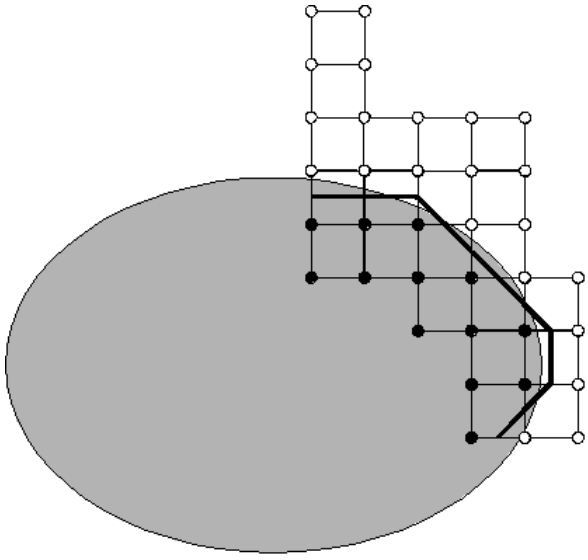

Figura 20: *Marching cubes* representado bidimensionalmente (*marching squares*).

Wilhelms e Gelder (1992) propuseram um método utilizando *octrees* para aumentar o desempenho na extração de superfícies, utilizando o mesmo princípio de ativação de vértices. Zhang, Bajaj e Sohn (2005), também utilizando *octrees* e uma variação do *marching cubes*, apresentaram um método para extração de superfícies adaptadas à simulação de sólidos. Kobbelt et al. (2001) apresentam uma versão estendida do *marching cubes* mais sensível a detalhes. Além destes, Guéziec e Hummel (1994) apresentaram o *Wrapper* e Kobbelt et al. (2001) apresentaram o *SpiderWeb*. Ambos se baseiam na valoração dos vértices para detecção da isosuperfície, no entanto, não necessitam de uma tabela que indica como as faces devem ser organizadas.

Apesar de ser uma forma eficiente para extração de superfícies para representação de sólidos a partir de dados médicos tridimensionais, o *marching cubes* pode gerar dados incorretos, como mostra a figura 21, onde dois sólidos diferentes foram fundidos em um único. Nielson e Hamann (1991), Matveyev (1994), Lewiner et al. (2003), Lopes e Brodie (2003) e Nielson (2003) propuseram métodos para corrigir tais ambigüidades.

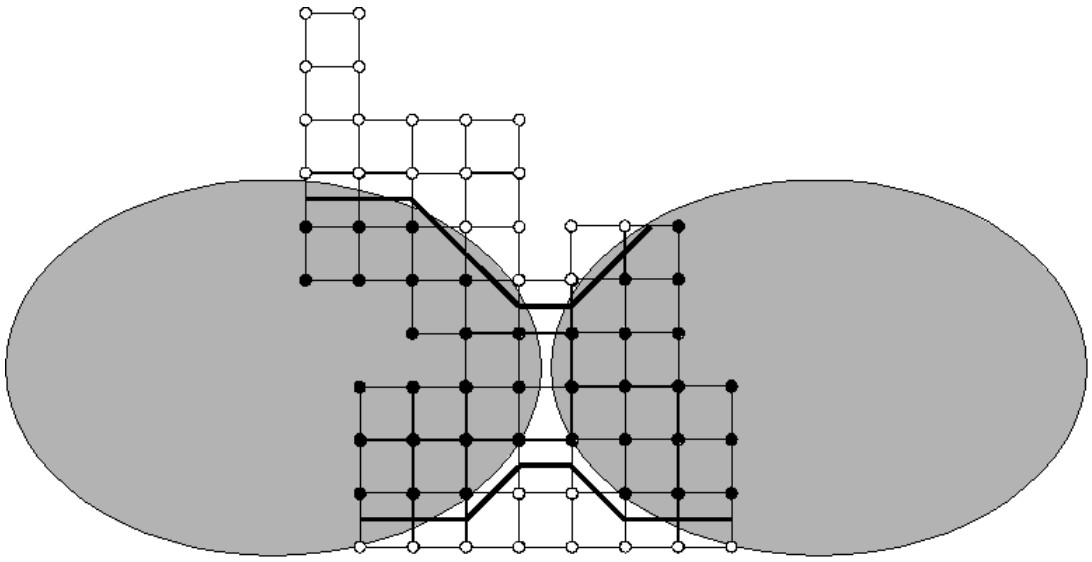

Figura 21: Erro na geração de sólidos utilizando o *marching cubes*, onde dois sólidos distintos foram fundidos em um único.

O resultado produzido pelo *marching cubes* é uma malha triangular densa, muitas vezes redundante e passível de redução sem perdas. Schroeder, Zarge e Lorensen (1992) desenvolveram um método para redução da malha, denominado decimação, o qual foi posteriormente otimizado por Boada e Navazo (2001), através da utilização de *octrees*. Murali e Funkhouser (1997) ap-

resentam um método para geração de representação por bordas a partir de polígonos arbitrários (como é o caso dos resultados do *marching cubes*), o qual é efetivo mesmo quando existe a intersecção entre polígonos, sobreposição, orientações erradas e outros problemas. Para a geração de representação de bordas utilizando polígonos com organização correta e orientação arbitrária, Krysl e Ortiz (2001) apresentam métodos para reorientação dos polígonos, geração de listas de faces, vértices e arestas e relações entre elas. Srinivasan, Akleman e Keyser (2004) demonstraram um método para geração de uma DLFL a partir de polígonos arbitrários, no entanto, caso o conjunto de polígonos não forme uma malha com propriedade *2-manifold* são introduzidas deformações nas faces para que esta característica seja mantida (ver seção 2.2.2.1 e capítulo 4). Um método para obtenção das orientações adequadas das normais das faces é apresentado por Borodin, Zachmann e Klein (2004).

Outra forma de visualização de superfícies, conhecida como *Ray Tracing* (LEVOY, 1988, 1990), é bastante utilizada devido a qualidade da imagem gerada. No entanto, esta técnica inviabiliza a simulação do sólido, estágio vital para a simulação cirúrgica (ver seção 2.1), pois não existe a geração de uma estrutura que represente o sólido. O resultado desta técnica é um conjunto de *voxels*. Além do *ray tracing* e das técnicas baseadas em isosuperfície, pode-se empregar técnicas de processamento de imagens para reconhecimento de bordas e técnicas de triangulações, como a de Delaunay, para a obtenção da representação de sólido, no entanto, estas geram resultados aproximados aos dos baseados em isosuperfície e perdem em desempenho.

A utilização de métodos baseados em isosuperfícies, apesar de bastante eficiente e eficaz na geração de malhas com qualidade elevada, não permite a variação rápida do nível de detalhamento da malha. Este nível é vinculado ao tamanho dos cubos utilizados na execução do método no caso dos métodos derivados do *marching cubes*. Para variar o nível de detalhamento é necessário que o método seja executado novamente. Outra forma de se obter níveis variáveis de detalhamento são apresentadas por Zorin, Schröder e Sweldens (1996), Kanai (2001) e Ma et al. (2002) onde, a partir de simplificações da malha original, a qual é muito complexa, obtém-se um ajustamento desta a uma malha de controle para superfície de subdivisão (figura 22). Labsik et al. (2002) demonstra como realizar a aquisição de dados em vários níveis de detalhes

utilizando o *marching cubes*.

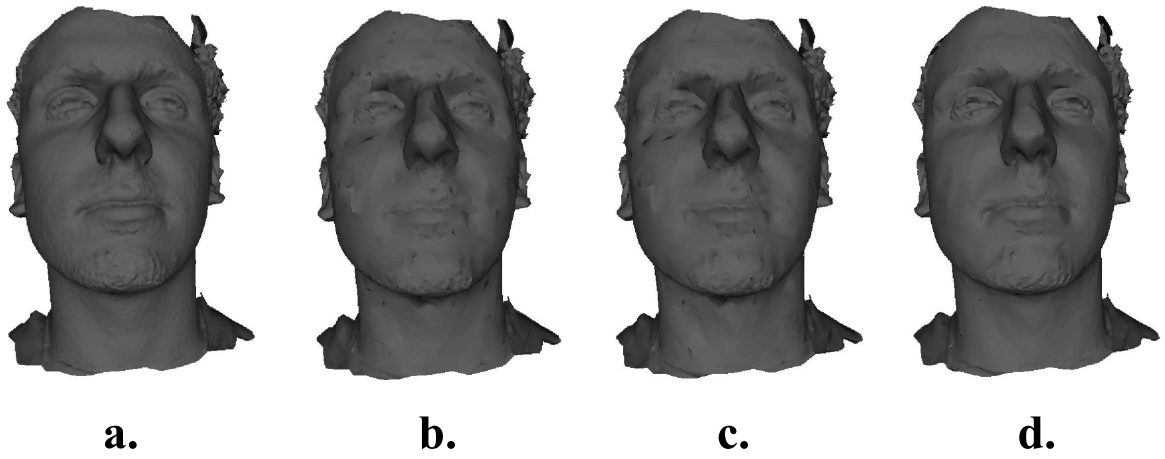

Figura 22: Adaptação de uma malha densa a uma superfície de subdivisão: a. malha original; b. malha decimada, com 10% do tamanho original; c. malha decimada ajustada a uma superfície de controle; d. malha resultante de uma subdivisão na malha de controle.

3 *Simulação Cirúrgica Imersiva*

Segundo Satava (1993, 1994), existem cinco elementos que afetam o realismo em um ambiente virtual para aplicações médicas (tabela 2 na página 17). Robb e Cameron (1994, 1995) ainda especificam que a taxa de atualização de vídeo deve se manter em torno de trinta *frames* por segundo e o tempo de resposta a uma ação do usuário deve ser menor que cem milisegundos (tabela 3 na página 18).

Neste capítulo serão analisadas as dependências computacionais para obtenção destes elementos e especificações e os requisitos para satisfação destes, sendo organizado da seguinte forma:

1. Apontamento das dependências computacionais para satisfação dos elementos definidos por Satava (1993, 1994) e as inter-relações entre elas (seção 3.1);

2. Apontamento das técnicas necessárias para satisfazer os requisitos apresentados por Robb e Cameron (1994, 1995), enquadrando-as no cenário definido na seção 3.1. Através deste enquadramento será demonstrado um modelo geral para simulação cirúrgica imersiva, indicando modelos e métodos envolvidos no processo (seção 3.2).

3.1 Dependências

Serão apresentados, individualmente, os elementos definidos por Satava (1993, 1994) e suas dependências computacionais, finalizando com a apresentação das inter-relações entre as dependências e os elementos. Ao final, as dependências identificadas para cada um dos elementos serão unificadas em um modelo único.

Nos diagramas apresentados nas seções seguintes utiliza-se a seguinte convenção:

- Caixas retangulares representam modelos estruturais (estruturas de dados, por exemplo);
- Caixas ovais representam modelos procedimentais (métodos e algoritmos, por exemplo);
- Setas contínuas representam dependência;
- Setas tracejadas representam dependência ocasional, a qual pode existir em determinadas aplicações, porém não em outras.

3.1.1 Fidelidade

A fidelidade, conforme explicitado por Satava (1993, 1994), é obtida através da utilização de gráficos de alta resolução. Para a obtenção deste nível de resolução deve-se, portanto, garantir detalhismo durante todo o processo de geração dos modelos para representação de sólidos, os quais serão posteriormente renderizados. Além disso, é necessário que os dispositivos de saída visual possam renderizar os dados com precisão. A partir destes pontos pode-se definir dois tipos de fidelidade: a fidelidade estrutural, a qual diz respeito a grau de semelhança entre a o modelo geométrico e o objeto real, e a fidelidade visual, definindo o grau de fidelidade entre o que é efetivamente desenhado e o objeto visual.

A fidelidade estrutural é importante para a obtenção de simulações numéricas realistas, no entanto a fidelidade visual não necessariamente precisa estar no mesmo nível da fidelidade estrutural. O nível de detalhismo necessário pode ser estipulado através de testes para medição de sensibilidade visual. Os testes de sensibilidade visual visam detectar as capacidades visuais do usuário, através de testes de percepção, modificando o nível de detalhe do modelo de acordo com as necessidades daquele (LUEBKE et al., 2002).

Sendo o processo de geração do modelo composto da seguinte forma (ver seção 2.1):

1. Aquisição de dados médicos tridimensionais;
2. Segmentação;

3. Geração de modelos para representação dos sólidos;

deve-se garantir que estes estágios estejam adequados com a fidelidade desejada. Além disso, os modelos devem ser capazes de representar tal fidelidade utilizando os recursos disponíveis. A figura 23 demonstra os estágios que influenciam na obtenção da fidelidade.

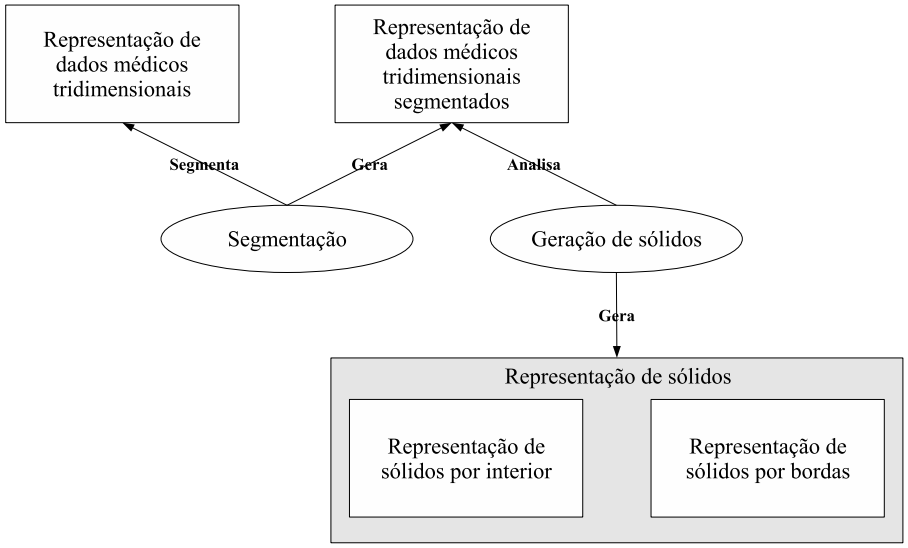

Figura 23: Processos e estruturas que influenciam na fidelidade.

3.1.2 Exposição de Propriedades dos Órgãos

A exposição de propriedades dos órgãos é realizada através de modelos para simulação de sólidos (ver seção 2.1). Tais modelos dependem de informações morfológicas e físicas do sólido, armazenadas em modelos para representação por bordas e por interior.

Modelos para representação de sólidos por bordas são utilizados para o armazenamento das massas e as constantes de mola, em modelos para simulação de sólidos como o massa-mola, enquanto modelos para representação por interior, como as *octrees*, são comumente utilizados para gerações de malhas adaptativas para elementos finitos (HO-LE, 1988). Durante a simulação das propriedades dos órgãos através de métodos de simulação, é necessário que haja bastante informação relacional entre os componentes da malha, pois as forças aplicadas em um componente são propagadas aos outros componentes. As estruturas de dados para representação de sólidos são as grandes responsáveis por manter estas informações.

Para que a simulação de sólido seja realizada de forma adequada, os componentes da malha que representa o sólido a ser simulado devem estar dispostos de forma a permitir a movimentação correta das partes que compõem o sólido. Por exemplo, a figura 24 mostra a aplicação de uma determinada força sobre malhas com diferentes níveis de detalhe. A aplicação da força em uma malha menos detalhada permite movimentos largos enquanto que a malha mais detalhada permite movimentos curtos e com maior nível de flexibilidade quanto ao formato final desejado. Dependendo do tipo de sólido que se deseja simular, requer-se mais ou menos detalhismo na malha. Como a maioria dos métodos para geração de malhas a partir de dados médicos tridimensionais geram malhas uniformes (ver seção 2.3), o nível de detalhismo das malhas é acertado através de métodos de adaptação de malha (MCRAE, 2001). Segundo McRae (2001), os objetivos primários da adaptação de malhas são redução do erro da discretização espacial e da dependência da malha final, visando a redução da propagação de erros numéricos durante o processamento.

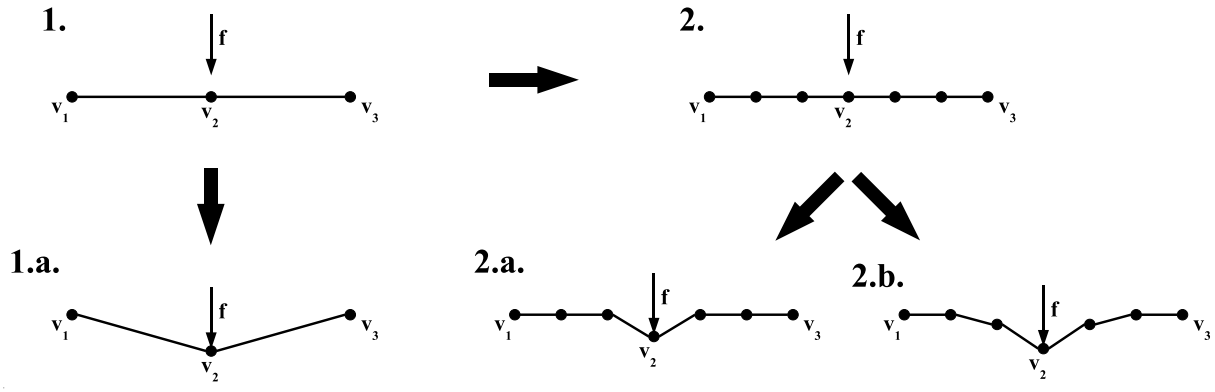

Figura 24: Manipulação de uma superfície utilizando malhas com diferentes níveis de detalhamento.

A adaptação da malha é realizada de forma numérica iterativa, baseada na aproximação dos resultados que seriam obtidos através do cálculo direto da deformação utilizando as informações relacionadas as propriedades físicas do sólido. Ainda, McRae (2001) especifica as seguintes propriedades que devem ser encontradas em métodos de adaptação de malhas:

- A adaptação de resultar em uma melhoria de precisão quantificável;
- A precisão temporal e a conservação devem ser preservados, se necessário para a apli-

cação;

- Erros adicionais introduzidos pelo método de adaptação não deve reduzir significativamente os benefícios;
- A adaptação deve ser automática e eficiente.

Métodos para adaptação de malhas são propostos por Sheffer e Üngör (2001), Huang (2001), George, Borouchaki e Laug (2002), Tristano et al. (2003) e Kallinderis e Kavouklis (2005). Baker (2005) propôs um método de adaptação de malhas que com modificação temporal e Frey (2004) propôs métodos para a adaptação de malhas anatômicas.

Reações como a cinemática das juntas requerem um modelo para representação de juntas e acoplamentos entre sólidos. Tal modelo é apresentado por Yamagushi e Kimura (1995) e Luo e Lukács (1991). Maciel, Nedel e Freitas (2002) apresentam um modelo para simulação de juntas anatômicas, onde se permite definir limites de rotações para os eixos. A figura 25 apresenta as estruturas e métodos envolvidos no processo de exposição de propriedades dos órgãos. Pode-se observar que a manipulação dos modelos é feita indiretamente, através de operadores de manipulação que mantêm a consistência das estruturas.

3.1.3 Exposição de Reações dos Órgãos

Satava (1993) exemplificou as reações dos órgãos através do vazamento de sangue de uma artéria ou bile do fígado. Este tipo de simulação é realizada através da inserção de fluidos dentro de cavidades internas dos sólidos, as quais são definidas em estruturas para representação de sólidos por interior. Os fluidos, por não possuírem forma definida e estarem, geralmente, em constante modificação desta, requerem uma estrutura de representação capaz de armazenar as mais variadas características das formas, sejam estas *2-manifolds*, *2-manifolds* com bordas ou até mesmo os *non-manifolds*.

A simulação dos fluidos é realizada através da modelagem de sistemas de interação entre partículas (REEVES, 1983a, 1983b; BURG, 2000; MÜLLER; CHARYPAR; GROSS, 2003) ou

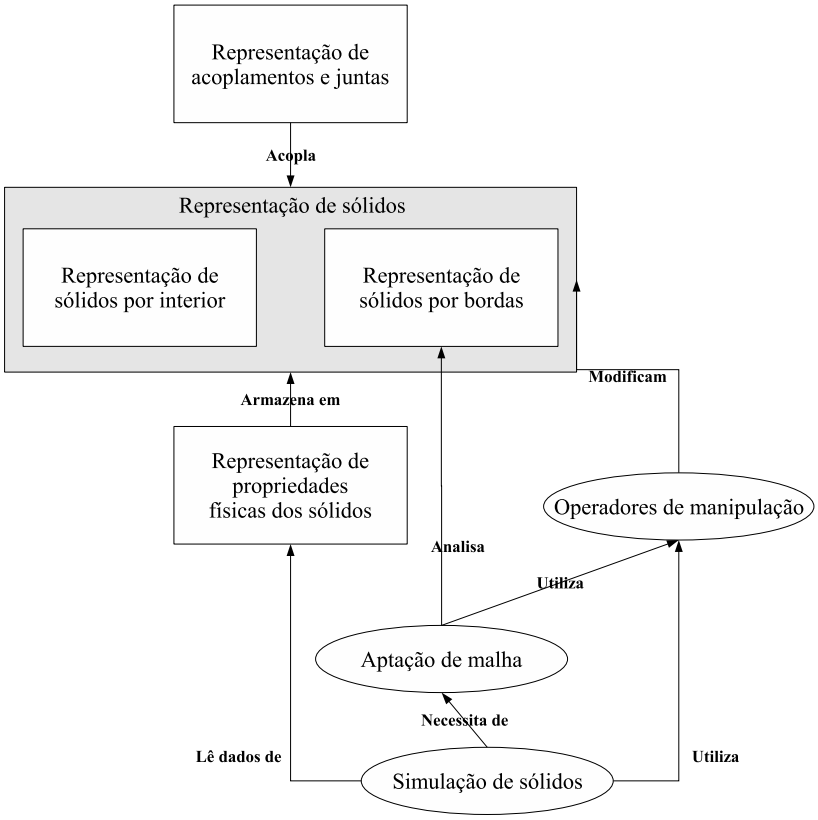

Figura 25: Processos e estruturas que atuam na exposição de propriedades dos órgãos.

através de simulação numérica (POZRIKIDIS, 2001). A junção de sistemas de partículas com sistemas de molas deu origem ao método para simulação de sólidos conhecido como massa-mola (BOURG, 2001; EBERLY, 2003), no entanto outros modelos para simulação de sólidos utilizando sistemas de partículas foram apresentados (SZELISKI; TONNESEN, 1992; WU; THALMANN; THALMANN, 1995; JAILLET; SHARIAT; VANDORPE, 1997). Sistemas de partículas são independentes, de forma que os métodos desenvolvidos para sua manipulação e renderização são voltados diretamente a estes sistemas, já que a forma clássica de renderização de objetos, utilizando polígonos, não é eficiente para estas estruturas (FOURNIER; FUSSELL; CARPENTER, 1982; REEVES, 1983a; REEVES; BLAU, 1985). No entanto, Müller, Charypar e Gross (2003) apresentaram um método eficiente que utiliza a geração de uma superfície para representação das partículas, permitindo que métodos comuns de renderização sejam aplicados, e gerando uma dependência ocasional entre o sistema de partículas e os modelos de representação por bordas.

Outra forma de reação encontrada são os movimentos musculares involuntários, tais como os batimentos cardíacos. Este tipo de reação é exposta através da utilização de modelos de

animação de sólidos. Watt e Watt (1992) e Hahn (1988) apresentam modelos para animação dinâmica (influenciada pela ocorrência de eventos) e que levam em consideração as propriedades físicas dos sólidos e descrevem modelos para animação realista. Sistemas de partículas são utilizados para animação de fluidos.

A figura 26 apresenta os processos e estruturas que influenciam na exposição de propriedades dos órgãos.

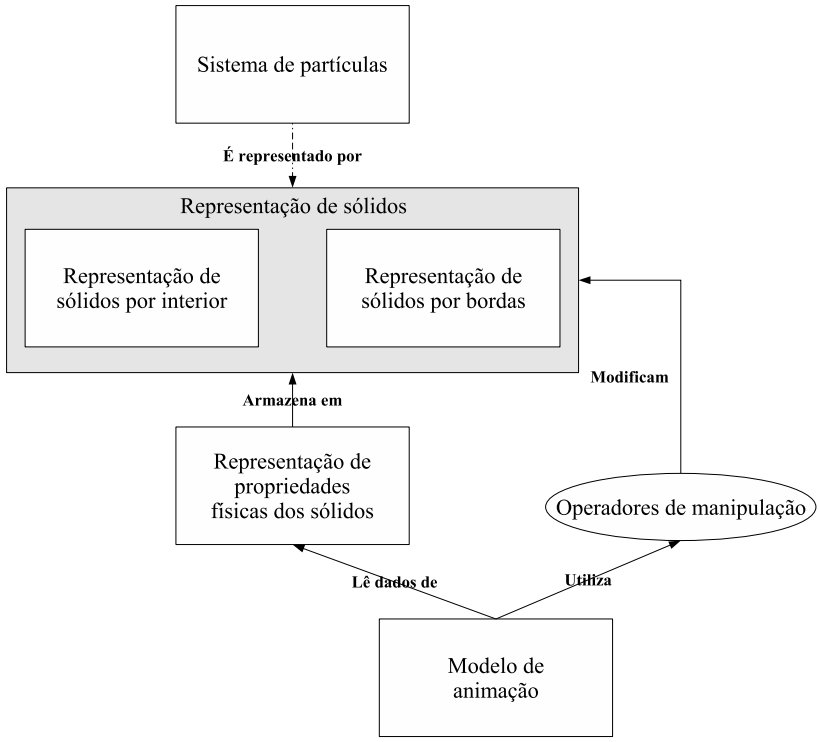

Figura 26: Processos e estruturas que influenciam na exposição de reações dos órgãos.

3.1.4 Interatividade entre Objetos e Retro-propagação Sensorial

A interatividade entre os objetos está diretamente ligada à detecção de colisão entre eles (ver seção 2.1). No entanto, o resultado da interação depende da aplicação: pode ocorrer uma deformação, através da simulação de sólido, ou a remoção de uma parte do sólido, através da utilização dos operadores de manipulação, por exemplo. Dependendo da aplicação e dos objetos que estão envolvidos na interação, pode-se determinar a realização de operações distintas. Estas operações estão vinculadas à aplicação a que se destina o ambiente virtual. No entanto, todas envolvem detecção de colisão entre objetos, visando determinar o contato entre eles.

Retro-propagação sensorial depende de características do ambiente e dos sólidos e de equipamentos que estimulem os órgãos sensoriais do usuário. Retro-propagação de força (tato) é descrita por Sheridan (1992), Mark et al. (1996), Dionisio et al. (1997), Dachille et al. (1999) e Lin e Otaduy (2005). Este tipo de simulação se baseia nas propriedades físicas do sólido para estipular a quantidade de retro-propagação necessária e, como baseia-se em contato (tato) entre usuário e objeto, existe também a dependência entre a detecção de colisão.

A figura 27 demonstra os processos e estruturas que influenciam na interação entre os objetos e na retro-propagação sensorial.

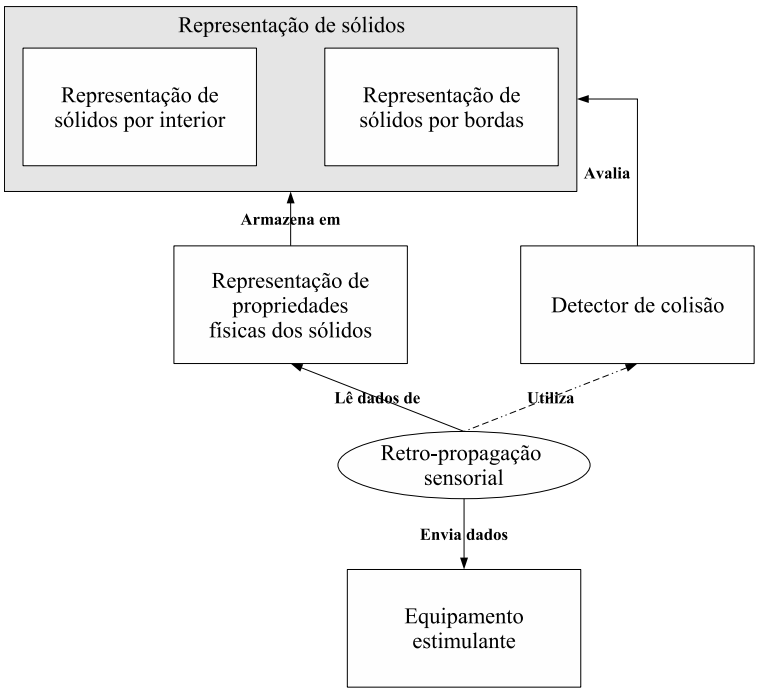

Figura 27: Processos e estruturas que influenciam na interação entre os objetos e na retro-propagação sensorial.

3.1.5 Modelo Único de Dependências

A figura 28 demonstra a unificação das dependências computacionais para satisfação dos elementos indicados por Satava (1993, 1994) em um único modelo. A representação dos sólidos se encontra no centro do modelo, sendo a única dependência computacional para todos os elementos. Este tipo de modelo, centrado na representação de sólido, é abordado por Mäntylä e Takala (1981) e Mäntylä e Sulonen (1982) e se enquadra inerentemente aos sistemas para mod-

elagem de ambientes virtuais para planejamento cirúrgico, o que pode ser constatado através da análise de dependências.

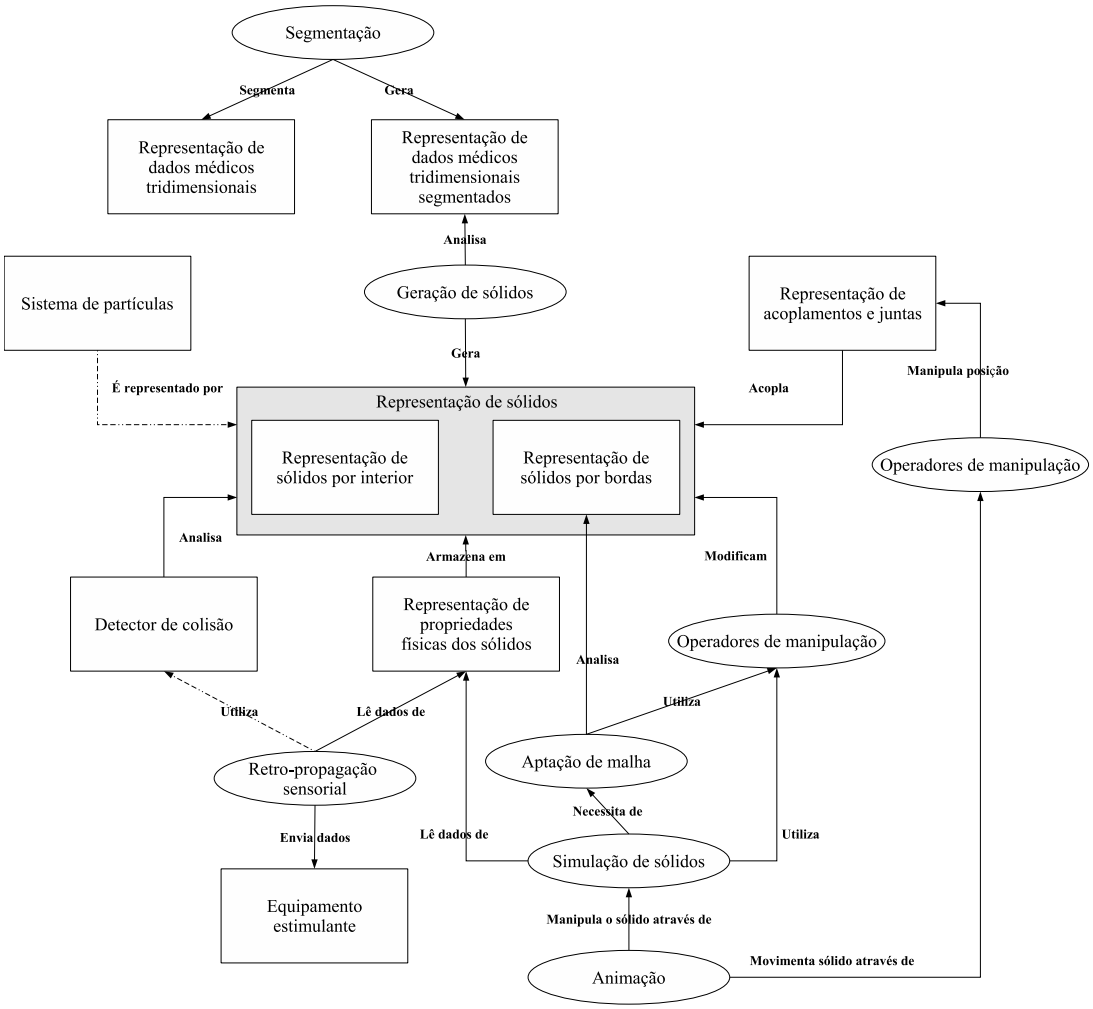

Figura 28: Modelo unificado das dependências computacionais para satisfação dos elementos explicitados por Satava (1993, 1994).

Na unificação das dependências, a animação deixa de ser um modelo estrutural e passa a ser procedimental. Esta mudança se deve ao fato de que as estruturas que possibilitam a animação do sólido passam a ser tratadas pelo modelo de juntas e acoplamentos, no caso da animação de movimentos (movimentos dos braços e pernas em relação ao corpo, por exemplo), e pelos métodos de simulação de sólidos, no caso de animações morfológicas (batimentos cardíacos, por exemplo). O modelo procedimental de animação passa a representar um *script* dos movimentos a serem realizados, tanto estática quanto dinamicamente.

3.2 Modelo Geral para Simulação Cirúrgica Imersiva

Apesar do modelo apresentado na seção 3.1 satisfazer as dependências computacionais para obtenção dos elementos apresentados por Satava (1993, 1994), estes não garantem que os requisitos apontados por Robb e Cameron (1994, 1995) sejam cumpridos. Embora o *hardware* computacional desempenhe papel decisivo na obtenção destes requisitos, técnicas de computação gráfica em tempo-real podem ser empregadas para atingir os requisitos e expandir a gama de equipamentos capazes de executar o modelo. Métodos numéricos para resolução de cálculos relacionados à geometria e física envolvida nos modelos garantem o desempenho destes e precisão adequada para aplicações médicas.

As técnicas de computação gráfica em tempo-real visam obter redução na quantidade de dados a ser submetidas a processos de renderização e simplificação em procedimentos realizados freqüente e corriqueiramente. Estas técnicas introduzem ao modelo uma quantia de métodos e estruturas que aumenta a necessidade de recursos para a execução daquele. No entanto, a execução do modelo sem estas técnicas não propicia o desempenho requerido por aplicações médicas de simulação em tempo-real devido à complexidade dos modelos para representação de sólidos (grande quantidades de dados para representar com precisão e fidelidade adequada para aplicações médicas) e ao desempenho dos métodos envolvidos na interação (como a detecção de colisão, por exemplo, onde testes de intersecção devem ser realizados entre uma quantidade vasta de dados).

Para eliminar grandes porções de dados do processo de renderização são aplicadas as técnicas de *culling*. A utilização de *culling* visa eliminar porções de objetos, possivelmente objetos inteiros, que não são vistos a partir do ponto de vista do observador (usuário) (EBERLY, 2000). As técnicas de *culling* são classificadas da seguinte maneira (MÖLLER; HAINES, 1999):

- *Culling* de volume de visualização: Visa eliminar de futuros processamentos de renderização objetos totalmente fora do volume de visualização do observador. Objetos que estiverem dentro ou parcialmente dentro do volume são submetidos ao processo de *clipping* (que visa identificar os limites do objeto com o volume de visualização). A figura

29 mostra um cilindro fora do volume de renderização, o qual não precisa ser submetido ao processo de sombreamento e renderização;

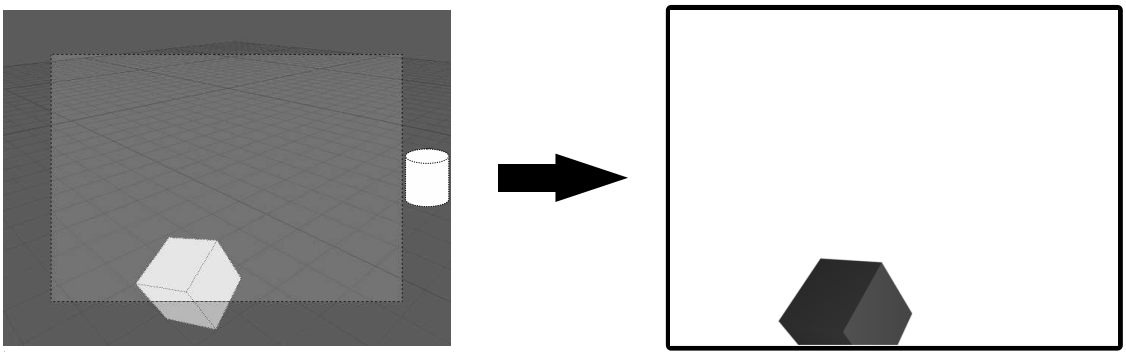

Figura 29: Volume de visualização visto de frente (a partir do ponto de vista do observador) e o resultado obtido após a renderização.

- *Culling* de faces ocultas: Visa eliminar as faces que estão do lado do objeto não enxergado pelo observador, as quais não precisam ser processadas pelo renderizador. A figura 30 mostra um cubo desenhado em formato de arame, onde são vistas todas as faces, e o resultado após o processo de renderização, onde são visíveis apenas as faces 1, 2 e 5;

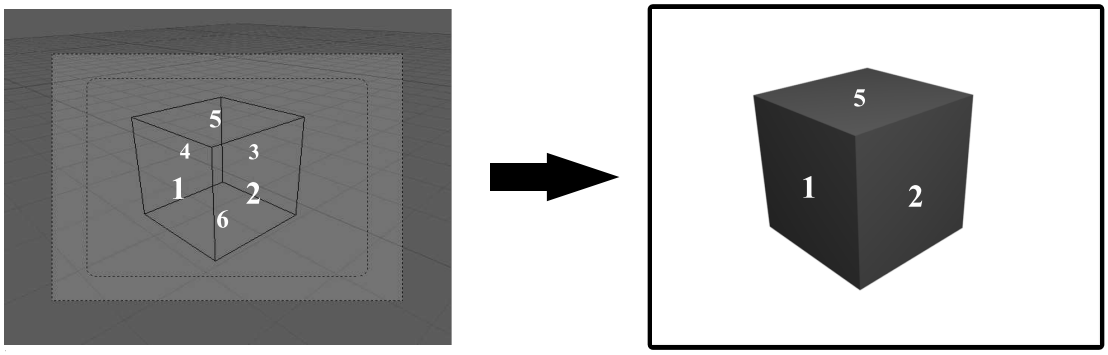

Figura 30: Cubo visto em arame visto de frente (a partir do ponto de vista do observador) e o resultado obtido após a renderização.

- *Culling* de oclusão: Visa eliminar de futuros processamentos de renderização objetos que não são visíveis do ponto de vista do observador devido estarem oclusos por outros objetos. Na figura 31 é mostrado o resultado de uma renderização onde um cubo está posicionado entre o observador e uma esfera, a qual não precisa ser submetida a futuros

processamentos;

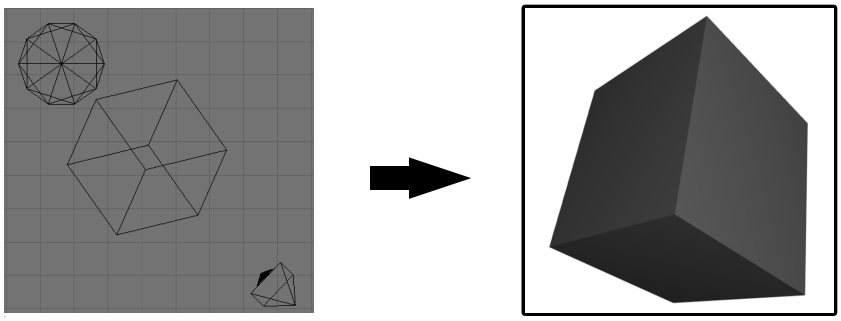

Figura 31: Cubo posicionado entre uma esfera e o observador e o resultado da renderização.

- *Culling* de portal: Este tipo de *culling* é uma forma de *culling* de oclusão, porém visa eliminar de futuros processamentos de renderização de regiões inteiras (EBERLY, 2004). Esta técnica se baseia na idéia de que a única forma de se observar objetos em um determinado ambiente a partir de outro ambiente é através de portais. A aplicação desta técnica é freqüente em aplicações com ambientes fechados, onde cada ambiente é conectado à outro através de portas, as quais desempenham o papel de portais. Em aplicações médicas, cortes na superfície de uma determinada estrutura abrem um portal da região exterior para a interior, permitindo que novos objetos sejam vistos.

Apesar de eliminar grandes porções de objetos e partes destes, as técnicas de *culling* introduzem processamento extra, o qual, ao invés de agilizar o processo, pode gerar atrasos. Isto pode ocorrer, por exemplo, na aplicação de *culling* de volume de visualização em conjuntos de objetos complexos e com vários componentes geométricos. Testes de intersecção devem ser realizados entre os planos do volume de visualização e os vários polígonos que compõem os objetos. Para agilizar este processo costuma-se representar os objetos por volumes de contorno mais simples (figura 32), como esferas ou paralelepípedos, e realizar os testes nestes volumes. Os métodos de construção de volumes de contorno são apresentados por Ritter (1990), Hearn e Baker (1996), Lengyel (2001) e Schneider e Eberly (2002). Ainda, para agilizar o processo, armazena-se estes volumes em estruturas hierárquicas para representação de objetos por interior, como as *octrees* (ver seção 2.2.1), permitindo, então, que os testes necessários sejam

realizados de forma hierárquica (FOLEY et al., 1990; MÖLLER; HAINES, 1999; EBERLY, 2000; LAMOTHE, 2003; EBERLY, 2004). Como um ambiente interativo está em constante modificação, estas estruturas devem ser capazes de ser manipuladas em tempo-real (LIBES, 1991; SHAGAM; JR., 2003a, 2003b; EPPSTEIN; GOODRICH; SUN, 2005; LARSSON; AKENINE-MÖLLER, 2005).

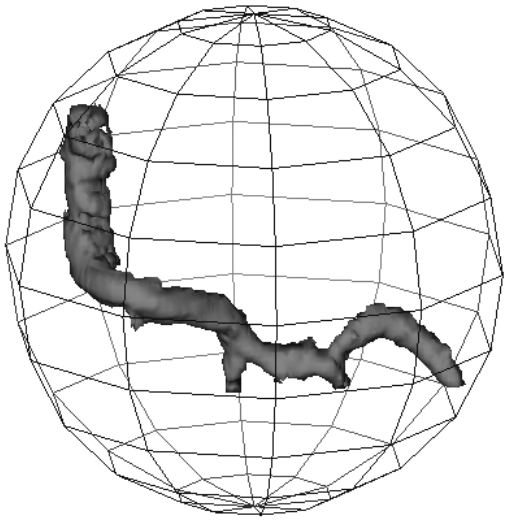

Figura 32: Esfera de contorno de uma artéria.

A utilização de volumes de contorno e armazenamento destes através de estruturas hierárquicas para representação de objetos por interior ainda é utilizada para agilizar outros processos, como a detecção de colisão (BERGEN, 2003). Neste caso, ao invés de ocorrer o teste de intersecção entre todas as faces de um determinado objeto com todas as faces de todos os outros objetos, testa-se apenas aqueles cujos volumes de contorno se intersectam.

Outra forma de se reduzir o número de elementos a serem processados é a utilização de superfícies de subdivisão (ver seção 2.2.2). As superfícies de subdivisão operam em vários níveis de detalhamento, possibilitando que em *hardwares* computacionais com maior capacidade de processamento este nível seja elevado, enquanto equipamentos com menor capacidade de processamento utilizem níveis menores. Este nível de detalhamento pode ser medido de acordo com as necessidades visuais do usuário, reduzindo a fidelidade visual caso o nível de detalhamento atual não seja necessário (LUEBKE et al., 2002) (ver seção 3.1.1).

Através da inserção dos métodos e estruturas apresentados nesta seção, os quais visam garantir a propriedade de tempo-real na aplicação e satisfazer os requisitos indicados por Robb e

Cameron (1994, 1995), no modelo apresentado na seção 3.1, obtém-se o modelo apresentado na figura 33. Este modelo satisfaz as dependências computacionais para obtenção dos elementos apresentados por Satava (1993, 1994) e garante os requisitos indicados por Robb e Cameron (1994, 1995).

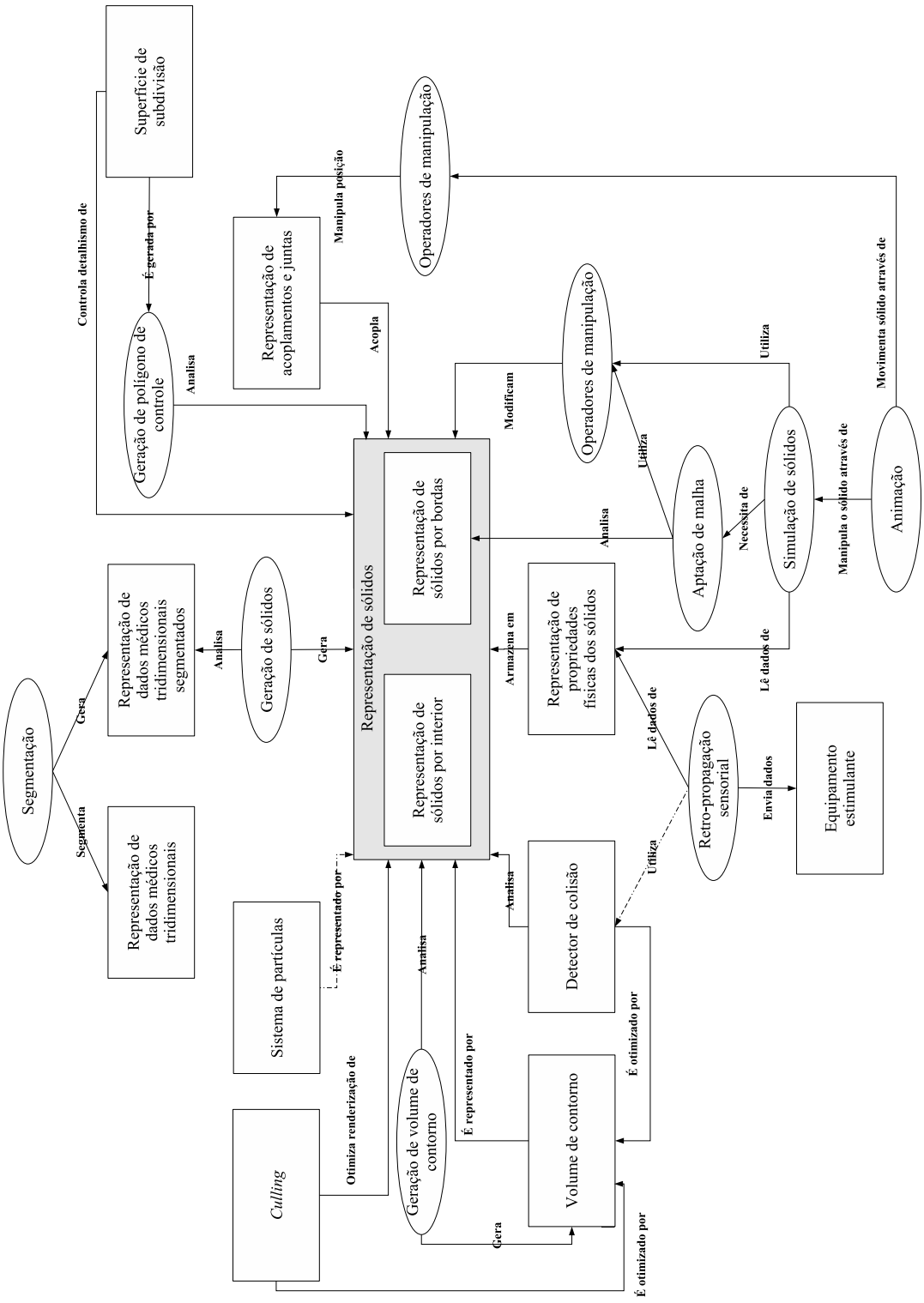

Figura 33: Modelo geral para simulação cirúrgica imersiva.

4 *Estendendo a DLFL*

Neste capítulo será abordada a *Doubly Linked Face List* (DLFL) (AKLEMAN; CHEN, 1999, 2000), suas limitações e desvantagens e duas extensões para esta estrutura de forma a permitir a representação de malhas *2-manifold* com bordas e malhas *non-manifold*, o que não se é capaz de obter com a DLFL original sem a ocorrência de deformação em algumas faces, visando manter as propriedades topológicas.

As duas extensões apresentadas não inviabilizam a utilização dos operadores originais (apresentados na seção 2.2.2.1) e não apresentam modificações estruturais significativas. A modificação da DLFL se dá, com uma pequena exceção, através da adição e da modificação dos operadores. Embora sejam apresentadas separadamente, as extensões podem ser utilizadas juntas em uma única estrutura.

Este capítulo está estruturado da seguinte forma:

1. Apresentação das limitações da DLFL, as quais pretende-se tratar através da adição de extensões à estrutura original (seção 4.1);

2. Apresentação das extensões para que seja possível a representação, através de uma DLFL, de malhas com propriedade *2-manifolds*, porém que possuem arestas sem identificação no sentido oposto, ou seja, uma malha com bordas (seção 4.2);

3. Apresentação das extensões para que seja possível a representação, através de uma DLFL, de malhas sem a propriedade *2-manifold* (*non-manifolds*) eliminando a degeneração de faces para garantir propriedades topológicas (seção 4.3);

4. Apresentação de um método de construção para DLFL a partir de polígonos arbitrários,

capaz de selecionar quais extensões propostas são mais adequadas aos dados a serem representados (seção 4.4).

A adição de representação de malhas *2-manifold* com bordas e de malhas *non-manifold* à DLFL resulta em uma estrutura baseada em faces, ideal para aplicações de interação entre usuário e sólidos (ver seção 2.2.2.1), com a capacidade de representar uma gama maior de sólidos. As operações de conjunto entre sólidos são extremamente propícias à geração de sólidos *non-manifold*, enquanto que formas geométricas com buracos nos locais das faces são facilmente obtidas através da representação de *2-manifolds* com bordas.

4.1 Limitações da DLFL

A primeira limitação da DLFL se encontra na tentativa de criação de um sólido aberto, como um cálice, por exemplo (figura 34). Esta limitação se deve à incapacidade da DLFL de representar bordas em malhas.

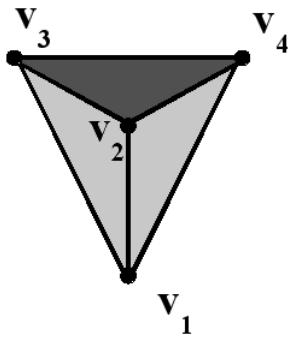

Figura 34: Um cálice formado por um tetraedro aberto no topo.

Para ilustrar esta incapacidade da DLFL será demonstrado passo a passo a tentativa de construir um cálice, através da utilização dos operadores desta estrutura. A figura 35 demonstra cada um dos estágios, apresentando o estado do objeto e do conjunto de faces, os quais são explicados a seguir:

1. Quatro inserções de vértices através do operador *CreateVertex*;

2. Inserção de aresta entre v_1 (f_1) e v_2 (f_2) através do operador *InsertEdge*, resultando na união destes dois vértices em uma mesma face;

3. Inserção de aresta entre v_1 (f_1) e v_3 (f_3) através do operador *InsertEdge*, resultando na união das duas faces envolvidas em uma única face e repetindo v_1 para manter a rotação nesta e a identificação dupla da nova aresta;

4. Inserção de aresta entre v_3 (f_1) e v_2 (f_1) através do operador *InsertEdge*, resultando na divisão da face original em duas faces em sentidos opostos, mantendo a propriedade *2-manifold*;

5. Inserção de aresta entre v_1 (f_2) e v_4 (f_4) através do operador *InsertEdge*, resultando na união das duas faces envolvidas em uma única face e repetindo v_1 para manter a rotação nesta e a identificação dupla da nova aresta;

6. Inserção de aresta entre v_2 (f_2) e v_4 (f_2) através do operador *InsertEdge*, resultando na divisão da face original em duas faces, onde f_2 representa as bordas do poliedro;

7. Inserção de aresta entre v_4 (f_2) e v_3 (f_2) através do operador *InsertEdge*, resultando na divisão da face original em duas faces;

8. O resultado das operações é um tetraedro fechado, com quatro faces triangulares.

Em uma estrutura *half-edge*, o próximo passo para a formação do cálice seria a aplicação do operador KFMRH (ver seção 2.2.2), no entanto, a DLFL não possuí tal operador. Além disso, a decisão de quais extremidades devem ser envolvidas na inserção de arestas é um processo heurístico. Por exemplo, caso se tivesse selecionado o vértice v_4 (f_3) e v_3 (f_1), o resultado seria a junção das duas faces, no entanto, ao renderizar-se as duas faces resultantes obter-se-ia o mesmo resultado visual (figura 36).

A tentativa de inserção de malhas *non-manifold* gera degenerações similares nas faces. A figura 37 ilustra a inserção de um novo tetraedro que compartilha um vértice com aquele apresentado na figura 35, da seguinte forma:

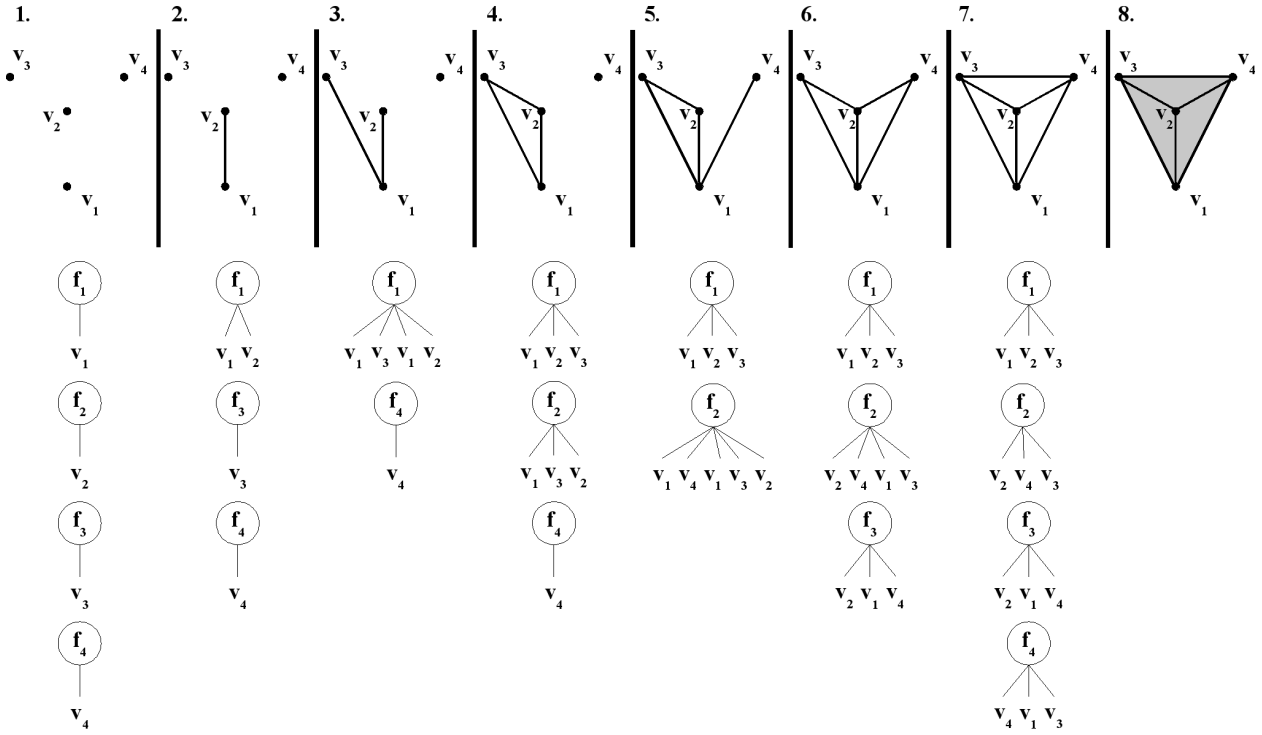

Figura 35: Tentativa de inserção de um cálice, formado por um tetraedro aberto no topo, em uma DLFL resulta em um tetraedro fechado.

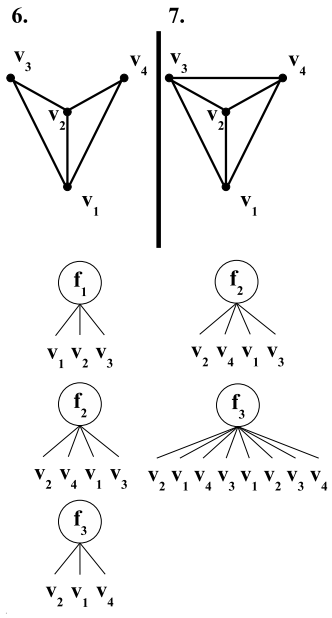

Figura 36: Inserção de aresta entre v_4 (f_3) e v_3 (f_1) no passo 7 da figura 35.

1. Estado inicial da malha, composta por um tetraedro e duas faces isoladas, as quais serão unidas

2. Inserção de aresta entre o vértice v_1 (f_1) e v_5 (f_6), através do operador *InsertEdge*. Várias opções de extremidades de aresta terminadas em v_1 são possíveis, no entanto, a seleção de qual delas é heurística, podendo-se optar por qualquer uma. O resultado desta operação é a junção das duas faces em uma única, repetindo os vértices entre os quais foi inserida a nova aresta para manter a rotação correta e a identificação dupla da nova aresta;

3. Inserção de aresta entre o vértice v_1 (f_1) e v_7 (f_1), resultando na da face em duas novas;

4. Inserção de aresta entre o vértice v_1 (f_1) e v_6 (f_1), resultando na da face em duas novas. Observa-se que, após terminada a construção do poliedro, a face f_1 representa uma rotação que poderia ser dividida em duas faces, porém, eliminando a propriedade *2-manifold*, pois a topologia em v_1 não seria homeomorfa a um disco aberto (ver seção 2.2.2).

Estas degenerações das faces, na tentativa de se representar uma gama maior de topologias, não é apropriada em aplicações em que se depende do conhecimento exato das relações entre os componentes, como no caso da simulação de tecidos, por exemplo. Tais degenerações podem ser eliminadas através de extensões, propostas nas seções seguintes.

4.2 Representando *2-manifolds* com bordas

A representação de malhas *2-manifold* com bordas é feita através das propriedades de inserção de arestas da DLFL e da atribuição de rótulos. Observando a figura 35 na página anterior, observa-se que a inserção de arestas sempre garante que todas elas possuem uma idêntica no sentido oposto através da repetição de vértices e da inclusão da faces com rotações opostas. No caso 6, da mesma figura, onde duas faces foram inseridas, existe uma terceira face que representa as bordas das outras duas, garantindo, desta forma, a propriedade *2-manifold* da estrutura como um todo. A atribuição de rótulos às faces que representam as bordas, e o gerenciamento

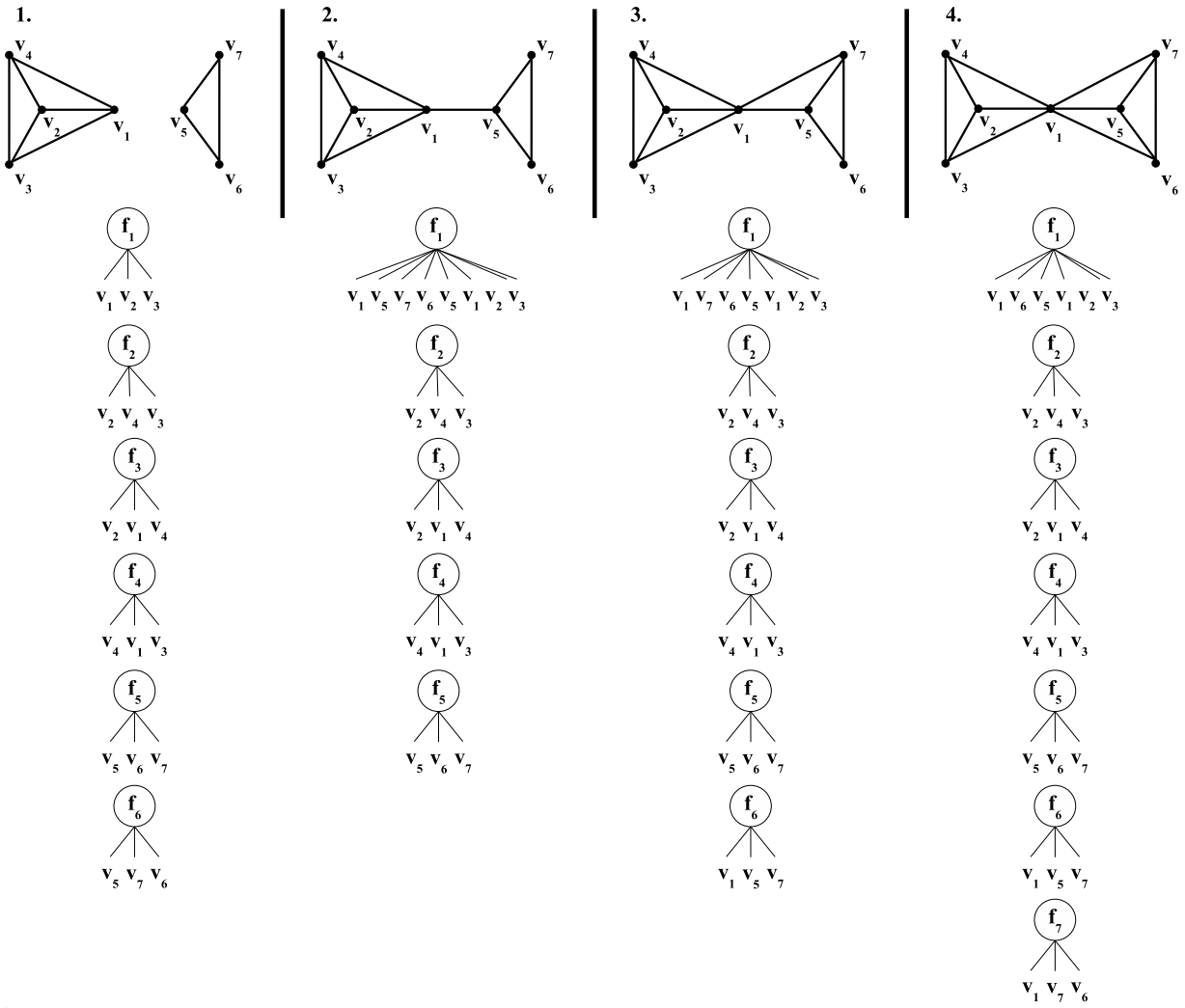

Figura 37: Inserção de uma malha *non-manifold* em uma DLFL.

destes, permite que estruturas *2-manifold* com bordas sejam inseridas na DLFL sem que a estrutura seja modificada. Esta rotulação permite que o cálice aberto (figura 34 na página 66) seja representado e que seja diminuída a quantidade de heurística durante a seleção de extremos para a inserção de arestas (conforme demonstra a figura 36 na página 68).

Conforme visto na seção 4.1, a inserção de uma aresta pode gerar a repetição de vértices e a união entre faces. Este procedimento visa manter a topologia *2-manifold* e a rotação contínua nas faces. No entanto, por se basear em um processo heurístico, a seleção de extremos pode gerar faces com organizações degeneradas (conforme figura 36 na página 68). Para evitar estas degenerações, uma boa heurística é a inserção de arestas a partir de uma única face, realizando a união das faces que contém os extremos desejados em uma única, e depois realizar uma inserção de aresta entre os dois membros desta face, fechando uma face nova, gerando uma divisão. O resultado da utilização desta heurística é uma face fechada conforme especificada pelo usuário e uma face que representa as bordas para manter a topologia *2-manifold*. Esta heurística foi utilizada na figura 36 (no caso 6, uma das faces representa as bordas) e não foi utilizada no caso demonstrado pela figura 36.

Seguindo o mesmo princípio da heurística mencionada, a expansão do modelo deve, sempre que possível, utilizar os extremos pertencentes às faces que representam bordas para inserção de arestas, evitando que faces corretas se degenerem. No entanto, durante um processo de várias inserções de arestas e junções e separações de faces, não se sabe qual face é desejada e qual delas representam as bordas. Para que se mantenha o rastreamento das faces que representam as bordas, pode-se utilizar um método de rotulação das faces e realizar pequenas modificações nos operadores de inserção e remoção de arestas, para que mantenham o rótulo atualizado.

Para a rotulação das faces é apresentado o operador *Bounding(Face f, Booleano is_bounding)*, o qual aplica o rótulo indicando representação de bordas na face f caso o parâmetro *is_bounding* seja verdadeiro, e remove o rótulo caso contrário. Além deste, o operador *IsBounding(Face f)* retorna verdadeiro caso a face f possua o rótulo de representação de bordas, e falso caso contrário.

O operador *InsertEdge* deve manter o rastro das faces que representam as bordas durante a junção e separação de faces. Adotou-se a seguinte estratégia:

- Durante a junção de faces: Caso alguma das faces envolvidas seja rotulada, a face resultante também é rotulada, visando não perder rótulos anteriores;
- Durante a quebra de uma face: Caso a face inicial seja rotulada, obtém-se uma face rotulada e uma não rotulada. Esta decisão é tomada através do sentido de inserção da aresta. A face resultante que mantiver os mesmos antecessores e sucessores dos extremos envolvidos na inserção de aresta herda o rótulo. Por exemplo, a inserção de aresta entre determinados vértices v_1 e v_2 e a inserção entre v_2 e v_1 geram faces rotuladas distintas.

O procedimento 1 apresenta o operador *InsertEdge* modificado. As letras de alfabeto grego representam seqüências de vértices não relevantes para o processo. O operador *InsertEdge* original e o procedimento *CoFacial*, utilizado por aquele, são descritos por Akleman e Chen (2000). No caso onde os extremos são co-faciais, e deve haver uma quebra das faces, optou-se por rotular a face que estiver no mesmo sentido da inserção da aresta ($f_1^{'}$), caso deseja-se rotular a face que possuir o sentido oposto ao da inserção da aresta, aplica-se o rótulo à face $f_1^{''}$. Este procedimento é utilizado para que não se perca o rastro da face rotulada. No entanto, caso a face rotulada não seja a face desejada, o rótulo é facilmente invertido. Ainda, o sentido para rotulação das faces pode ser parametrizado no operador.

A figura 38 mostra a inserção de um cálice da mesma forma demonstrada na figura 35 na página 68, no entanto, utilizando a extensão para representação e rastreamento das faces que representam bordas. Nesta figura optou-se por rotular uma face desde o início, no entanto, o mais indicado seria inserir o rótulo em uma das faces fechadas do passo 4. No final (passo 8) existe um cálice aberto, já que a face de cima é rotulada e pode ser interpretada como os anéis da estrutura *half-edge* (ver seção 2.2.2). Cada passo da imagem é descrito a seguir (arestas e faces em cinza são rotuladas):

1. Inserção dos vértices e rotulação de f_1;

Procedimento 1 Operador *InsertEdge* modificado (o "*" indica os passos inseridos).

INSERTEDGE(c_1, c_2)

1. Assume-se c_1 igual a COR(e_1, e_2) (ver seção 2.2.2.1), representando o vértice v_1, e c_2 igual a COR(e_3, e_4), representando o vértice v_2.
2. Se COFACIAL(c_1, c_2) então
   (a) Achar a face $f_1 = \alpha w_1 v_1 w_1' \beta w_2 v_2 w_2'$ contendo c_1 e c_2;
   (b) Quebrar a seqüência em w_1', obtendo as subseqüências $\alpha w_1 v_1$ e $\beta w_2 v_2 w_2'$;
   (c) Quebrar a subseqüência $\beta w_2 v_2 w_2'$ em w_2', obtendo $\beta w_2 v_2$;
   (d) Concatenar $\alpha w_1 v_1$, v_2 e w_2' para obter a face $f_1' = \alpha w_1 v_1 v_2 w_2'$;
   (e) Concatenar $\beta w_2 v_2$, v_1 e w_1' para obter a face $f_1'' = \beta w_2 v_2 v_1 w_1'$;
   (f) * Se ISBOUNDING(f_1) então BOUNDING(f_1', VERDADE);
3. Senão
   (a) Achar a face $f_1 = \alpha w_1 v_1 w_1'$ contendo c_1;
   (b) Achar a face $f_2 = \beta w_2 v_2 w_2'$ contendo c_2;
   (c) Quebrar a seqüência $\alpha w_1 v_1 w_1'$ em w_1', obtendo $\alpha w_1 v_1$;
   (d) Quebrar a seqüência $\beta w_2 v_2 w_2'$ em w_2', obtendo $\beta w_2 v_2$;
   (e) Concatenar $\alpha w_1 v_1$, v_2, w_2', $\beta w_2 v_2$, v_1 e w_1', obtendo $f_n = \alpha w_1 v_1 v_2 w_2' \beta w_2 v_2 v_1 w_1'$;
   (f) * Se ISBOUNDING(f_1) ou ISBOUNDING(f_2) então BOUNDING(f_n, VERDADE);

2. Inserção de aresta entre faces distintas faz com que elas sejam unificadas e, como uma delas é rotulada, a face resultante herda o rótulo;

3. Mesma ocorrência do passo 2;

4. A inserção de aresta entre v_3 e v_2 gera a subdivisão da face original, onde a nova face f_2 recebe o rótulo por possuir os extremos no mesmo sentido da inserção de aresta;

5. Junção de faces faz com que o rótulo seja herdado pela resultante;

6. A inserção de aresta entre v_2 e v_4 faz com que a face que representa as bordas contorne as arestas externas das duas outras faces;

7. A inserção de aresta entre v_4 e v_3 passa o rótulo para a face do topo;

8. Interpretando as faces rotuladas da mesma forma que um anel na *half-edge* pode-se obter uma estrutura aberta (com bordas) como este cálice tetraédrico.

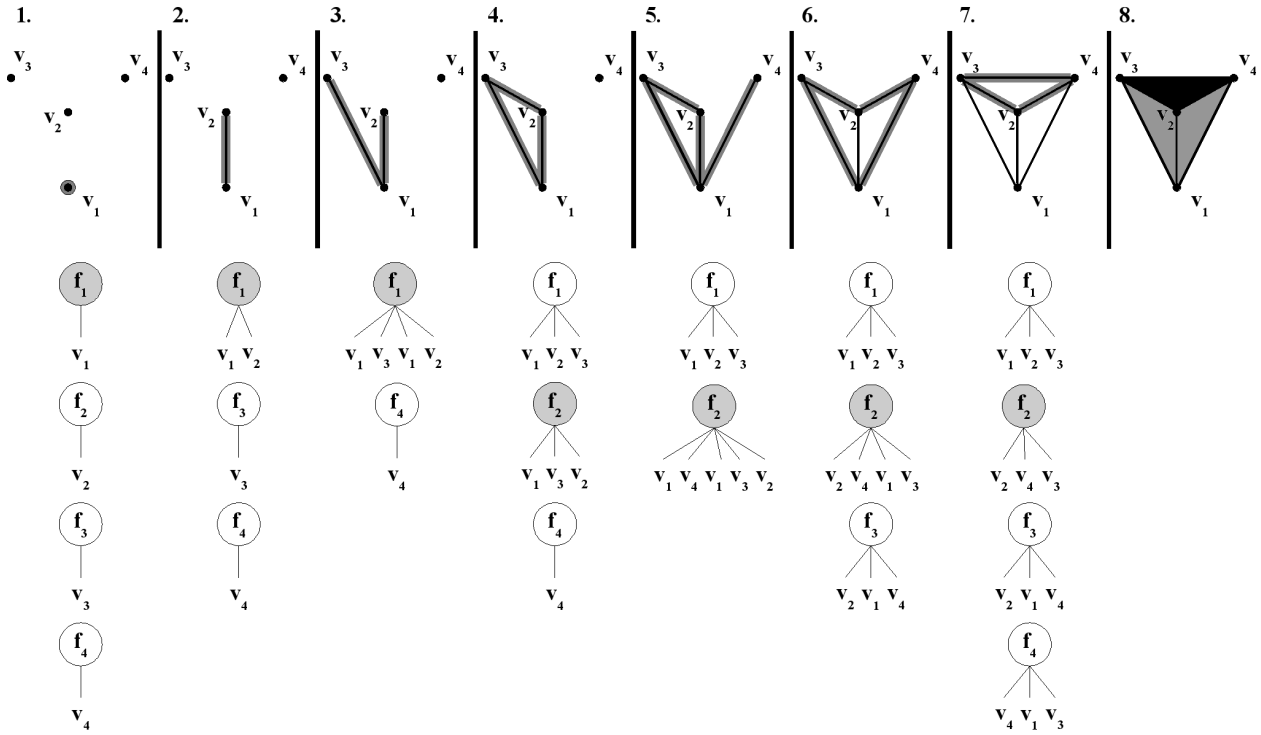

Figura 38: Inserção de um cálice tetraédrico em uma DLFL com extensão para representação de bordas.

A utilização de rotulação permite que seja eliminada a heurística do processo de seleção de extremos, mas introduz um processo heurístico de seleção de faces a serem rotuladas. Em uma

construção de polígonos a partir de vértices, o processo é bastante controlável e se adquire as faces desejadas sem muitas complicações. Durante a construção da DLFL a partir de polígonos arbitrários, degenerações podem surgir caso as faces que representam as bordas estejam organizadas em sentidos opostos. A figura 39 demonstra a inserção de arestas entre faces rotuladas quando estas estão orientadas no mesmo sentido (caso 1) e em sentidos opostos (caso 2).

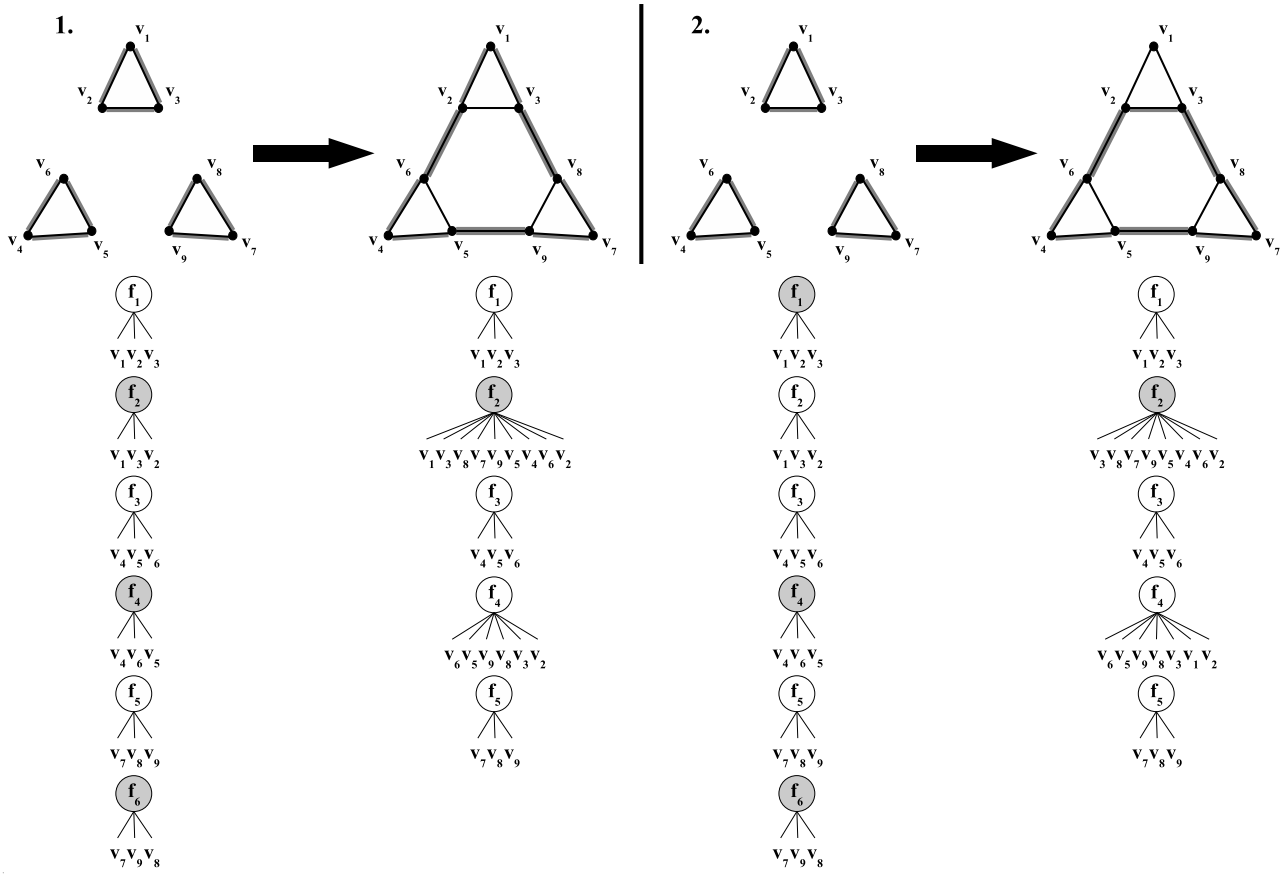

Figura 39: Inserções de arestas entre faces rotuladas com orientações no mesmo sentido e sentidos opostos.

Em ambos os casos da figura 39, a inserção de arestas ocorre na seguinte seqüência: (v_3, v_8), (v_9, v_5) e (v_6, v_2). No caso 1, onde as faces rotuladas possuem a mesma orientação, obtém-se uma face rotulada resultante que percorre todas as bordas da malha, enquanto que, no caso 2, a face rotulada resultante não cobre corretamente as bordas, devido a uma face rotulada inicial em orientação oposta às outras. Uma verificação das orientações das faces rotuladas pode ser aplicada antes do processo de inserção de arestas. Também, no caso 1, se as arestas fossem inseridas na ordem inversa, a rótulo seria herdado pela face interna, facilmente resolvido com uma troca de rótulos entre elas.

O operador de remoção de arestas, *DeleteEdge*, é modificado conforme mostra o procedimento 2. O operador original é descrito por Akleman e Chen (2000). Caso os dois lados da aresta pertençam à mesma face, e esta for rotulada, as duas faces resultantes herdam o rótulo. Isto ocorre pela impossibilidade de se concluir logicamente qual das faces resultantes passa a representar as bordas. Possivelmente, ambas as faces representam as bordas, caso a inserção da aresta, a qual está sendo removida, tenha sido realizada entre duas faces previamente rotuladas. Para evitar a perda do rastro do rótulo, optou-se por rotular ambas, cabendo ao usuário remover o rótulo de uma delas, caso pertinente. A remoção de aresta na qual um dos lados, ou ambos os lados, pertença a uma face rotulada, a face resultante herda o rótulo. A união das faces envolvidas faz com que o intervalo pertencente à face não rotulada se torne uma borda, conforme mostra a figura 40, onde a aresta e_1, pertencente a uma face rotulada e uma não rotulada, é removida, colocando nas bordas as outras arestas pertencentes à antiga face f_1.

Procedimento 2 Operador *DeleteEdge* modificado (o "*" indicam os passos inseridos).

DELETEEDGE(e)

1. Achar os dois lados idênticos (v_1, v_2) e (v_2, v_1) de e;
2. Se os lados de e pertencem a diferentes faces $f_1 = \alpha v_1 v_2$ e $f_2 = \beta v_2 v_1$ então
   (a) Quebrar a seqüência $\alpha v_1 v_2$ em v_1 para obter α;
   (b) Quebrar a seqüência $\beta v_2 v_1$ em v_2 para obter β;
   (c) Concatenar α, v_1, β e v_2 para obter $f_n = \alpha v_1 \beta v_2$;
   (d) * Se ISBOUNDING(f_1) ou ISBOUNDING(f_2) então BOUNDING(f_n, VERDADE);
3. Senão \\Os dois lados de e pertencem a mesma face $f_1 = \delta v_1 v_2 \gamma v_2 v_1$
   (a) Quebrar a seqüência $\delta v_1 v_2 \gamma v_2 v_1$ de forma a obter $f_1' = \delta v_1$ e $f_1'' = \gamma v_2$;
   (b) * Se ISBOUNDING(f_1) então
      i. * BOUNDING(f_1', VERDADE);
      ii. * BOUNDING(f_1'', VERDADE);

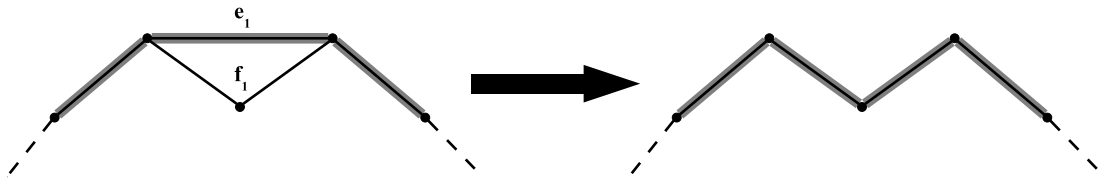

Figura 40: Remoção de uma aresta cujo um dos lados pertence a uma face rotulada.

4.3 Representando *non-manifolds*

Normalmente, a representação de malhas *non-manifold* elimina várias vantagens topológicas e geométricas que são garantidas pela propriedade *2-manifold*, como, por exemplo, a organização regular e circular das faces, as rotações únicas e unidirecionais a partir dos vértices e a facilidade de obtenção de informações relacionais entre os componentes da malha. Para garantir estas e outras vantagens, as estruturas de representação se tornam bastante inflexíveis e deixam a desejar em questões de desempenho de operações muitas vezes essenciais, como as operações de conjunto entre malhas, por exemplo, as quais podem gerar topologias *non-manifold*. Além disso, a modelagem de determinados sólidos através de malhas *2-manifold* pode se tornar complicada em alguns casos, já que o compartilhamento de arestas e vértices em rotações distintas não é possível.

Para manter as vantagens proporcionadas pela propriedade *2-manifold* e obter a flexibilidade das malhas *non-manifold,* Mäntylä (1988) propôs que a representação destas fosse feita através da junção de partes com propriedade *2-manifold*. Por exemplo, os tetraedros conectados por um único vértice, na figura 37 na página 70, podem ser representados por dois tetraedros distintos (duas partes com propriedade *2-manifold*) cuja conexão entre eles é exposta através de algum indicador de conexão dentro da estrutura. Desta forma toda a estrutura mantém as propriedades *2-manifold*, exceto os pontos de conexão. Além disso, os objetos dentro da mesma estrutura podem compartilhar a lista de vértices e arestas, reduzindo a ocupação de recursos de armazenamento, e são mantidas as propriedades de vizinhança.

Higashi et al. (1993) adotaram a idéia apresentada por Mäntylä, aplicando-a à *half-edge* e adicionando novos operadores, similares aos de Euler, para tratar os pontos onde não existe a propriedade *2-manifold*. Ainda, a equação de Euler-Poincaré foi estendida para validar tais

extensões *non-manifold*. Para a DLFL adota-se uma representação similar.

Conforme exposto na seção 2.2.2.1, a DLFL é a implementação de um sistema de rotação de grafo, o qual foi provado por Edmonds (apud AKLEMAN; CHEN, 1999) equivaler sempre a uma malha *2-manifold* orientada. Um sistema de rotação de um determinado grafo é uma lista de rotações, uma para cada vértice deste grafo, sendo uma rotação de um determinado vértice uma permutação cíclica das arestas incidentes à este. Como pode-se ver no tetraedro (figura 35 na página 68) cada vértice possuí uma única rotação, ao contrário dos tetraedros conectados por um vértice (figura 37 na página 70), onde o vértice que faz a junção entre os tetraedros possuí duas rotações distintas.

Na DLFL, a junção de componentes *2-manifold* se dará através da inserção de uma nova rotação para um determinado vértice. Para este fim, é apresentado o operador *AddRotation(*v*)*, o qual insere uma nova lista de rotações para o vértice v. Portanto, cada nodo da lista de vértices da DLFL, ao invés de conter uma lista de extremos de arestas, indicando uma rotação, passa a ser uma lista de listas, cada uma destas representando uma rotação distinta. Além de adicionar uma nova rotação a um determinado vértice v, o operador *AddRotation* ainda cria uma esfera pontual (uma face com um único vértice) centrada em v, da mesma forma que atua o operador *CreateVertex*. De forma resumida, o operador *AddRotation* tem o mesmo efeito da inserção de um mesmo vértice pela segunda vez, possuindo uma nova rotação. Esta forma geraria um *2-manifold* geométrico, já que o mesmo vértice existiria duas vezes, sendo uma nova entidade topológica, porém não haveria identificação entre os vértices, mantendo a topologia corretamente *2-manifold*. No entanto, a identificação entre os vértices é muitas vezes desejada, já que eles são geometricamente o mesmo, e a reinserção de vértices causaria um aumento de necessidade de armazenamento na lista de vértices. O operador *AddRotation* resolve os problemas de identificação entre vértices e o de ocupação de unidades de armazenamento.

A figura 41 demonstra a inserção dos dois tetraedros da figura 37 utilizando o operador *AddRotation*, mostrando, ainda, o estado do nodo da lista de vértices que representa o vértice v_1 (o qual serve de ponto de junção entre os dois tetraedros). A junção dos tetraedros procede da seguinte forma:

1. A primeira rotação da lista de rotações do nodo que representa o vértice v_1 mostra os três extremos de aresta representados por este nodo, todos fazendo parte da mesma porção *2-manifold;*

2. Execução do operador *AddRotation(v_1)*, resultando na adição de uma nova rotação na lista de rotações e criando uma esfera pontual centrada em v_1 e representada pela face f_7;

3. Inserção de aresta entre v_1 (f_7) e v_5 (f_6), resultando na junção das faces a qual pertencem;

4. Inserção de aresta entre v_1 (f_6) e v_7 (f_6), resultando na divisão da face a qual pertencem em duas faces distintas;

5. Inserção de aresta entre v_1 (f_6) e v_6 (f_6), resultando na divisão da face a qual pertencem e gerando o polígono final, contendo um vértice de junção com duas rotações e nenhuma face degenerada.

De forma oposta ao operador *AddRotation*, é introduzido o operador *CleanRotations(v_1)*, o qual remove da lista de rotações do vértice v_1 todas as rotações que representam esferas pontuais (rotações com um único vértice), exceto a primeira, caso todas as rotações representem esferas pontuais.

A inserção de arestas *non-manifold* (arestas que pertençam a mais de uma rotação de vértice), se dá através da inserção desta aresta entre dois vértices com rotação extra. Por exemplo, para adicionar uma aresta entre os vértices v_1 e v_2, sendo que já existe uma aresta entre eles, o procedimento a ser realizado é a execução do operador *AddRotation* sobre ambos os vértices e inserir uma aresta entre os novos extremos de aresta. No entanto, para haver a correspondência entre arestas geometricamente idênticas, deve-se buscar o nodo da lista de arestas que representa esta aresta, e adicionar um ponteiro para os extremos neste. Desta forma, cada nodo de aresta passa a ser uma lista de tuplas relacionando extremos de arestas. Para otimizar a busca do nodo que representa a aresta *non-manifold* a ser inserida, ao invés de se realizar comparações na lista de aresta, busca-se na rotação de um dos vértices envolvidos na operação, qual dos extremos está contido nesta aresta, e daí chega-se no nodo da lista de arestas através das faces.

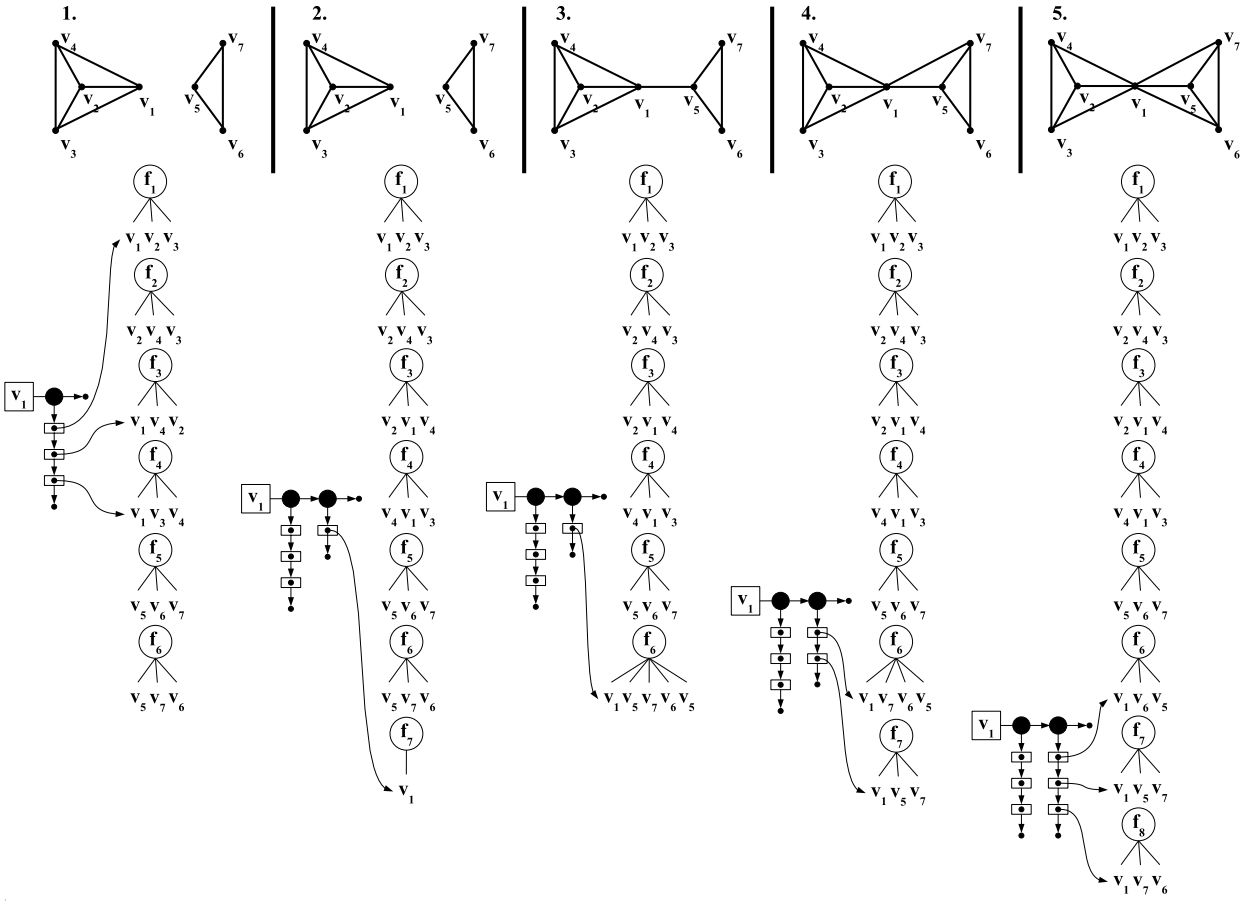

Figura 41: Inserção de dois tetraedros conectados por um vértice em uma DLFL estendida para representação de topologias *non-manifold*.

O processo de identificação entre faces requer a identificação de todos os membros que existem nas faces e, por isso, é um método de busca extensiva. Por esse motivo, a identificação de faces não é incorporada na estrutura. No entanto, modelagens com faces geométricas idênticas podem ser realizadas através de representações com propriedade *2-manifolds*, com remoção de uma das faces, e sem acarretar degenerações nas outras faces.

A estrutura da DLFL com extensão para representação de *non-manifolds* passa a ser definida como a tripla $< F, V, E >$, onde F representa uma lista de seqüências, cada seqüência representando uma face, V representa uma lista onde cada nodo representa um vértice e uma lista de rotações, sendo cada rotação uma lista de extremos de arestas, e E representa uma lista onde cada nodo representa uma aresta e uma lista de tuplas de extremos de arestas, onde cada tupla representa a mesma aresta em uma rotação distinta. A figura 42 mostra o estado da DLFL estendida para representação de *non-manifolds* durante a representação dos tetraedros unidos por um único vértice, mostrado na figura 41.

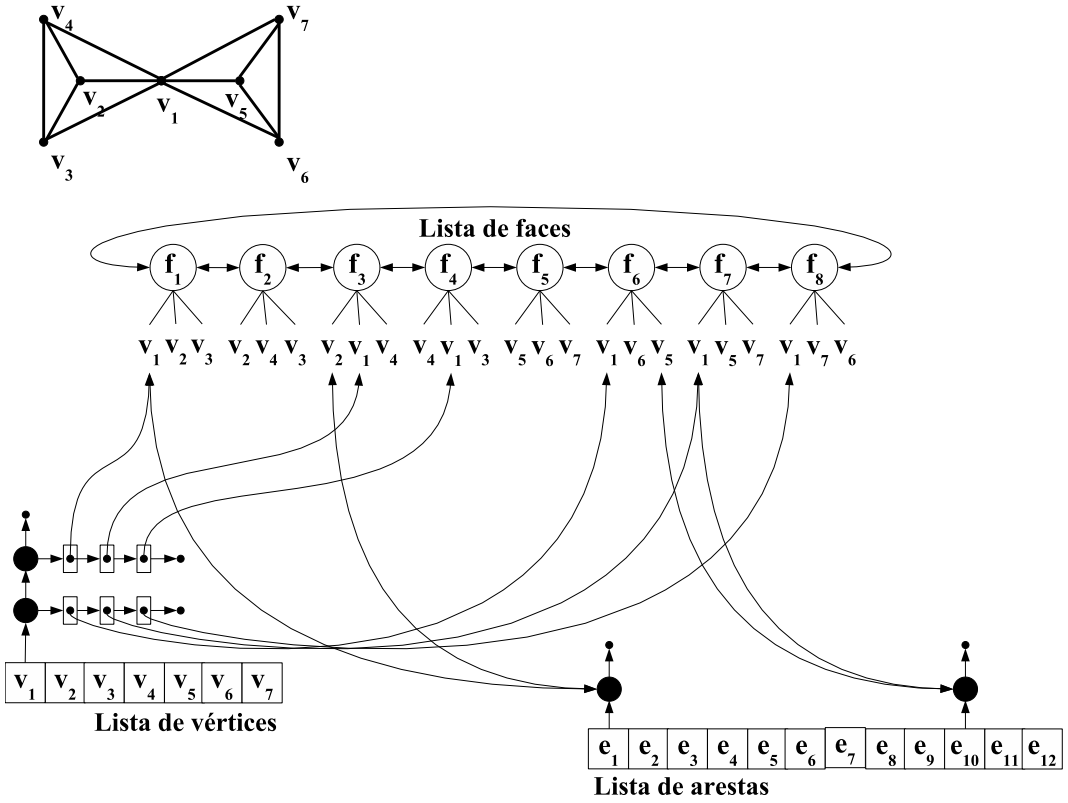

Figura 42: Estado da DLFL com extensão para *non-manifolds* durante a representação dos tetraedros unidos por um único vértice, mostrado na figura 41.

Os operadores de remoção e inserção de arestas mantém seu funcionamento normalmente,

podendo ser aplicados mesmo entre extremos de arestas presentes em partes *2-manifolds* distintas, já que não é necessário fazer a distinção entre as partes. As únicas partes que precisam ser distintas das outras são aquelas onde ocorrem os pontos de junção *non-manifold*.

4.4 Construindo a DLFL

A DLFL pode ser construída de duas formas: a inserção de vértices e arestas pelo usuário, utilizando os operadores da estrutura, ou através de malhas arbitrárias, geradas por um método externo, como, por exemplo, o *Marching Cubes* (ver seção 2.3). Durante a construção da DLFL pelo usuário, este pode decidir quais extensões utilizar, dependendo das circunstâncias em que a modelagem será aplicada. No entanto, a construção a partir de malhas arbitrárias requer que as extensões sejam decididas conforme as propriedades topológicas da malha e, preferencialmente, sem intervenção do usuário e sem degenerações das faces iniciais.

Krysl e Ortiz (2001) mostraram como transformar uma triangulação em uma estrutura para representação por bordas. É apresentado como verificar a propriedade *2-manifold* na malha triangular, orientar as faces, caso possível, e construir as listas de faces e arestas. Embora aplicados a malhas triangulares, estes métodos podem ser estendidos para malhas compostas por polígonos arbitrários. Importante notar que estes métodos se baseiam apenas nas propriedades topológicas da malha, ignorando as propriedades geométricas. Um exame sobre as ambigüidades geométricas das representações por bordas é demonstrado por Hoffmann e Hopcroft (1987) e Murali e Funkhouser (1997). No entanto, para se obter uma representação adequada da malha pela DLFL, a lista de vértices não deve ser ambígua, para que as propriedades de vizinhança não sejam perdidas.

Neste trabalho assume-se que a malha de entrada informa a organização dos polígonos que compõem a malha de alguma forma, seja através de índices para uma lista de vértices ou de conjuntos de coordenadas indicando os vértices. Para casos onde se indica apenas a conexão entre os vértices e requer-se um processo de identificação de ciclos dentro deste conjunto de conexões, os quais formam as faces, indica-se o trabalho de Krysl e Ortiz (2001), que demonstram como

identificar os ciclos e obter uma organização que indique os componentes dos polígonos. Os métodos apresentados a seguir supõem a não existência de ciclos dentro das faces (subfaces nas faces).

Para que se dê início ao processo de construção da DLFL, os dados de entrada devem sofrer uma organização inicial. Esta organização visa a construção de uma lista de vértices não-ambígua, caso não exista, e a construção de uma lista de faces (polígonos) cujos componentes são indexados na lista de vértices. Após a etapa de organização das faces e vértices busca-se obter uma orientação única das faces. Durante a busca de uma orientação única, a qual se obtém através do percurso pelas arestas da estrutura, obtém-se também a divisão da malha inicial em partes *2-manifold* (e *2-manifold* com bordas). Durante a terceira etapa busca-se encontrar os componentes *non-manifold*, os quais atuam como pontos de junção entre partes *2-manifold* distintas. Na quarta etapa é realizado o processo de inclusão dos dados na DLFL. Nas seções adiante serão descritas cada uma das etapas mencionadas.

4.4.1 Organização Inicial

A organização inicial dos dados de entrada visa prover uma lista não-ambígua de vértices e uma lista de faces, cujos nodos de ambas as listas possuem conexões entre si. Estas conexões servem para a identificação de correlações entre os vértices e as faces, as quais serão necessárias nas próximas etapas. Estas listas de faces e vértices criadas neste momento não são as listas de faces e vértices da DLFL, já que estas ainda podem ser subdivididas durante as próximas etapas.

Nesta etapa de organização considera-se a pior organização dos dados de entrada possível, onde as faces são compostas por um conjunto seqüencial de coordenadas representando os vértices. A não ser que cada face seja por si só uma parte isolada, existe redundância entre os vértices. A redundância entre os vértices denota uma lista de vértices ambígua, devido ao fato de que a mesma posição geométrica é representada por diferentes entidades topológicas.

A obtenção de uma lista de vértices não-ambígua, caso esta não seja provida pela malha inicial, requer a aplicação de uma busca extensiva, comparando todos os vértices da lista com

todos os outros restantes. A utilização de árvores e tabelas de *hash* podem auxiliar no desempenho deste processo. Caso vértices sejam identificados como sendo o mesmo, as faces que incluem estes vértices devem apontar para o mesmo índice, e o vértice deve apontar para as faces em que está contido. Assim sendo, cada nodo da lista de faces é composto por uma lista de ponteiros para os nodos da lista de vértices e cada nodo da lista de vértices é composto por uma lista de ponteiros para os nodos da lista de faces.

O método para obter as listas de vértices e faces iniciais é mostrado pelo procedimento 3.

Procedimento 3 Método para obter a organização inicial da malha de entrada.

1. $V_i \leftarrow$ `nova lista de vértices;`
2. $F_i \leftarrow$ `nova lista de faces;`
3. `Para cada face` f_e `da malha de entrada fazer`
   (a) $f_i \leftarrow$ `nova face em` F_i`;`
   (b) `Para cada conjunto de coordenadas` c `de` f_e `fazer`
      i. `Buscar um vértice` v `em` V_i `que possua as mesmas coordenadas` c`;`
      ii. `Se não existe` v `então` $v \leftarrow$ `novo vértice em` V_i `com coordenadas` c`;`
      iii. `Se não existe ponteiro para` v `em` f_i `então inserir ponteiro para` v `em` f_i`;`
      iv. `Se não existe ponteiro para` f_i `em` v `então inserir ponteiro para` f_i `em` v`;`

4.4.2 Orientação das Faces e Divisão da Malha em Partes *2-manifold*

Após a obtenção de uma organização inicial da malha de entrada, a qual permite a identificação única dos vértices que compõem a malha e a quais faces cada vértice está associado, busca-se, nesta etapa, orientar as faces em um sentido único, quando possível, identificar os componentes *2-manifold* e *2-manifold* com bordas que compõem a malha. Nesta etapa detecta-se, também, a necessidade da utilização da extensão da DLFL para representação de estruturas *2-manifold* com bordas.

A organização das faces em um sentido único visa organizar a inserção de arestas na DLFL

de forma a não degenerar as faces, tanto em casos *2-manifold* e *2-manifold* com bordas (ver seções 4.1 e 4.2). Separar a malha inicial em partes *2-manifold* e *2-manifold* com bordas permite que a inserção delas na DLFL seja feita de forma separada, realizando a junção entre elas através da utilização dos operadores para representação de estruturas *non-manifold*, caso tais junções existam.

O procedimento 4 demonstra o método para obtenção dos elementos mencionados. Cada parte *2-manifold* é armazenada separadamente em uma lista de faces. Estas listas de faces são então armazenadas em uma lista (FF). Além disso há um rotulo para cada uma das partes *2-manifold* que indica a necessidade da utilização da extensão para *2-manifolds* com bordas. Este método segue as arestas de uma determinada face, buscando as faces adjacentes à aquela: caso exista apenas uma face adjacente a uma determinada aresta, esta é orientada conforme a primeira e incluída na mesma parte *2-manifold*; caso existam mais de uma face adjacente, encontrou-se um ponto *non-manifold*, o qual será rotulado na próxima etapa; caso não exista nenhuma face adjacente, encontrou-se uma borda, mostrando a necessidade da utilização da extensão para *2-manifolds* com bordas. Após buscar-se todas as faces adjacentes à uma determinada face, busca-se as adjacentes destas, até que a parte *2-manifold* seja totalmente identificada. Isto é realizado através da inserção das adjacentes em uma pilha (S). No momento em que a pilha estiver vazia, todas as faces que compõem uma determinada parte *2-manifold* foram encontradas, permitindo que se prossiga para a busca das outras partes, caso existam.

4.4.3 Detecção de Componentes *Non-manifold*

A detecção de componentes *non-manifold* se torna bastante simples após as etapas anteriores. Para descobrir-se os vértices *non-manifold*, basta-se encontrar os vértices que apontam para faces em partes *2-manifold* distintas. As arestas *non-manifold* são compostas por dois vértices *non-manifold* seqüenciais dentro de alguma face, os quais serão formados por conseqüência da inserção dos dados na DLFL, na próxima etapa. Os vértices pertencentes a mais de uma parte *2-manifold* atuam como pontos de junção entre as partes. Caso existam partes sem pontos de junção, estas estão isoladas, podendo ser representadas por DLFLs distintas, caso seja

Procedimento 4 Método para obter uma orientação única das faces, divisão da malha em partes *2-manifold* e detecção de componentes *non-manifold*.

1. $S \leftarrow$ nova pilha;
2. $FF \leftarrow$ nova lista de listas de faces;
3. $F \leftarrow$ nova lista de faces em FF;
4. Colocar no topo de S primeira face f_i de F_i;
5. Remover f_i de F_i;
6. Enquanto S não estiver vazia fazer
   (a) $f \leftarrow$ remover o topo de S;
   (b) Inserir f em F;
   (c) Para cada aresta e de f fazer
      i. Se e é *manifold* então
         A. $f_o \leftarrow$ face do lado oposto de e;
         B. Orientar f_o de acordo com f;
         C. Se f_o pertence à F_i então
            1. Colocar f_o no topo de S;
            2. Remover f_o de F_i;
      ii. Senão
         A. Se e existe em apenas uma face então rotular F como "COM_BORDAS";
   (d) Se S estiver vazia e F_i não estiver vazia então
      i. $F \leftarrow$ nova lista de faces em FF;
      ii. Colocar no topo de S primeira face f_i de F_i;
      iii. Remover f_i de F_i;

conveniente.

A tarefa de detectar os componentes *non-manifold* de uma malha de entrada arbitrária, a qual já foi tratada pelas etapas 1 e 2, é mostrada pelo procedimento 5. Através deste procedimento é indicada a necessidade da extensão para representação de malhas *non-manifold* da DLFL. Os vértices rotulados como *non_manifold* atuam como pontos de junção entre as partes. As partes que mantiverem o rótulo *manifold* até o final da etapa, estão isoladas de todas as outras partes.

Procedimento 5 Método para detecção dos componentes *non-manifold* da malha de entrada.

1. Rotular todos os vértices v de V_i como "MANIFOLD";
2. Rotular todas as partes F de FF como "MANIFOLD";
3. extensão_*non_manifold* ← FALSO;
4. Para cada vértice v de V_i fazer
   (a) F_p ← parte a qual pertence a primeira face f apontada por v;
   (b) *non_manifold* ← FALSO;
   (c) Para cada face f apontada por v fazer e
      i. F_f ← parte a qual pertence f;
      ii. Se não *non_manifold*
         A. Se F_f for diferente de F_p então
            1. *non_manifold* ← VERDADE;
            2. Rotular F_f como "NON-MANIFOLD";
      iii. Senão rotular F_f como "NON-MANIFOLD";
   (d) Se *non_manifold* então
      i. Rotular F_f como "NON-MANIFOLD";
      ii. Rotular v como "NON-MANIFOLD";
      iii. extensão_*non_manifold* ← VERDADE;

4.4.4 Inserção dos Dados na DLFL

De forma similar à etapa 1, onde são acertadas as rotações das faces e detectada as partes *2-manifold* através do rastreamento das faces adjacentes, o processo de inserção na DLFL também utiliza a busca de adjacências. Por este motivo, a criação de uma matriz de adjacências, já na

primeira etapa, agilizaria o processo nesta etapa.

Mesmo que não seja necessária a utilização da extensão para a representação de *2-manifolds* com bordas, esta é utilizada para auxiliar o processo de detecção dos pontos onde devem ser expandidas as faces. Para cada parte *2-manifold* estarão disponíveis um conjunto de esferas pontuais para a construção desta, sendo que para os vértices *non-manifold* uma nova esfera unitária será criada no início da inclusão de cada parte, através do operador *AddRotation*. Sendo assim, cada extremo de aresta da DLFL que faz parte da malha, estará representado por uma esfera pontual, ou estará presente em alguma face rotulada como representação de borda, caso o extremo já tenha sido envolvido alguma vez na inserção de uma aresta. O extremo permanecerá em uma face rotulada até que se feche o ciclo de faces em sua volta.

No final do procedimento, caso não seja necessária a utilização da extensão para a representação de *2-manifolds* com bordas, os rótulos são removidos de todas as faces, evitando a interpretação de buracos inexistentes na estrutura.

O procedimento 6 mostra o método de inserção dos dados na DLFL, sendo que estes já tenham sido processados pelas etapas 1, 2 e 3. Neste procedimento, considera-se o sentido de rotulação das faces no sentido contrário ao da inserção da aresta (ver seção 4.2).

Procedimento 6 Método para inserção dos dados na DLFL.

1. Para todos os vértices v de V_i fazer CREATEVERTEX(v);
2. Para cada parte F de FF fazer
   (a) Se extensão_*non_manifold* então adicionar uma rotação, através do operador ADDROTATION(v), para todos os vértices com rótulo "NON-MANIFOLD" apontados por qualquer face de F;
   (b) $S \leftarrow$ nova pilha;
   (c) Inserir no topo de S a primeira face de F;
   (d) $f_d \leftarrow$ face da lista de faces da DLFL F_d que contém o primeiro vértice de f;
   (e) BOUNDING(f_d, VERDADE);
   (f) Enquanto S não estiver vazia fazer
      i. $f \leftarrow$ remover o topo de S;
      ii. Para cada aresta e de f fazer \\Sendo e composto por v_1 e v_2 nesta ordem dentro de f.
         A. Buscar o extremo de aresta c_1 pertencente a f_{d_1} em F_d que representa $v_1$tal que f_{d_1} possua apenas um vértice ou ISBOUNDING(f_{d_1});
         B. Buscar o extremo de aresta c_2 pertencente a f_{d_2} em F_d que representa $v_2$tal que f_{d_2} possua apenas um vértice ou ISBOUNDING(f_{d_2});
         C. Se e é *manifold* então
            1. $f_o \leftarrow$ face do lado oposto de e;
            2. Inserir f_o no topo de S;
   (g) Se não F não possuí rótulo "COM_BORDAS" então
      i. Para cada face f_d de originada de F ISBOUNDING(f_d, FALSO);

5 *Resultados*

Utilizando as estruturas apresentadas e seus métodos de criação, foi desenvolvido um aplicativo para medição de artérias para customização de próteses. Este aplicativo mede o centro, raio médio e perímetro de polígonos originados através da intersecção entre um plano arbitrário e a DLFL. A artéria é segmentada através de *split & merge* (ver seção 2.2.1) e a malha é gerada através do *marching cubes* (ver seção 2.3). A figura 43 ilustra a intersecção entre dois planos e a artéria, evidenciando os polígonos gerados pela intersecção.

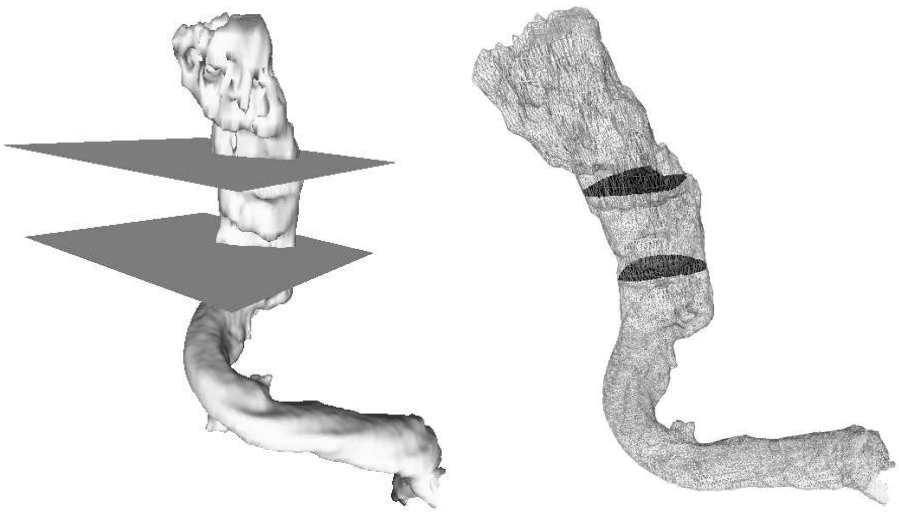

Figura 43: Intersecção de uma artéria por dois planos, evidenciando os polígonos formados pela intersecção.

O polígono através da estrutura é encontrado através de testes de intersecção entre as arestas o os planos arbitrários (O'ROURKE, 2001). A DLFL é bastante útil para a realização dos testes de intersecção. Supondo a não utilização desta estrutura, mas sim de uma lista de polígonos, todas as arestas da malha deveriam ser testadas, armazenando-se os pontos de intersecção. Após a detecção de todos os pontos de intersecção, estes devem ser organizados para que se obtenha

o polígono, já que não sabe-se *a priori* a ordem em que as arestas foram testadas. Utilizando-se a DLFL, basta encontrar, na lista de arestas, uma que intersecte o plano. Achando a intersecção de uma aresta, basta procurar as intersecções na face oposta, até que se feche o ciclo. Desta forma, não é necessário que todas as arestas da malha sejam testadas e a ordem dos pontos de intersecção são obtidas de forma direta.

Para a validação das medições foi realizado um experimento no qual se sabe as medidas do objeto real. Para isso, inseriu-se uma quantidade de material radiopaco dentro de três tubos plásticos, contorcendo-os logo após (figura 44). Uma seção destes tubos foi então tomografada e uma reconstrução tridimensional realizada (figura 45). Foram traçados quatro planos na mesma posição onde medições através de um paquímetro digital foram realizadas manualmente nos tubos. Os resultados referente ao diâmetro médio são mostrados na tabela 6, para cada um dos tubos.

Figura 44: Modelo com medidas conhecidas utilizado para validar as medições do software.

Há uma certa discrepância oscilando em torno de 1 mm entre os valores medidos pelo software e dos valores medidos manualmente. A causa disto se atribui ao fato de que cada *pixel* na imagem representa uma área em torno de 0.5 mm^2, e não um ponto geométrico exato. A transformação desta área em pontos geométricos, o que permite a renderização, leva em conta esta área na transformação, no entanto existe uma perda. Outro fator que influencia as medições manuais é a precisão da mão humana, a qual está sempre sujeita a erros.

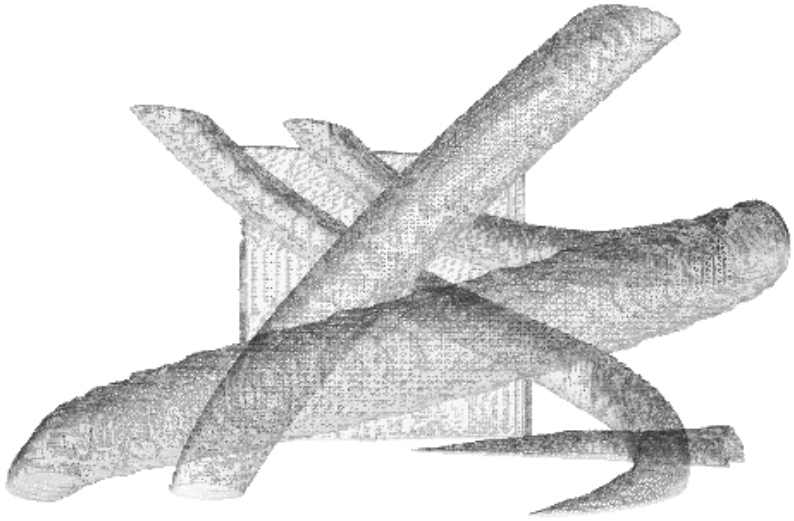

Figura 45: Reconstrução tridimensional do modelo apresentado na figura 44.

	Tubo 1		**Tubo 2**		**Tubo 3**	
	Manual	Software	Manual	Software	Manual	Software
Plano 1	25.86	25.41	24.65	27.59	25.21	25.54
Plano 2	25.6	26.11	25.63	24.75	25.26	24.94
Plano 3	25.38	26.47	24.97	24.8	25.46	25.21
Plano 4	23.79	26.74	23.71	25.26	24.55	25.8
Média	**25.16**	**26.18**	**24.74**	**25.6**	**25.12**	**25.37**

Tabela 6: Resultados das medições de diâmetro manuais e do software, em milímetros.

A medição do software é influenciada pela geração de erros de três processos: acquisição das imagens, segmentação e reconstrução. Os erros de aquisição é facilmente mensurável, já que, conforme mencionado, é informado na imagem (no caso da artéria, cada *pixel* equivale a 0.5 mm^2). No entanto, os erros de segmentação e reconstrução se tornam mais difíceis de mensurar, pois requerem a medida do objeto real (difícil de obter no caso de uma artéria, por exemplo). Além disso, a disassociação entre os dois erros baseando-se apenas na medida do objeto real e a medida do *software* é impossível. As verificações devem ocorrer em etapas, para que haja uma separação entre os erros.

6 *Conclusões*

Por se representar um conjunto de dados tridimensionais através de uma seqüencia de imagens bidimensionais, a análise de casos médicos que dependem deste tipo de informação se torna bastante inflexível. Técnicas de computação gráfica e realidade virtual podem ser utilizadas para suprir as inflexibilidades inerentes aos exames médicos comuns. Tais técnicas visam prover a capacidade de representação tridimensional dos dados médicos, visualização em tempo real e a manipulação destes de forma realista. Estas capacidades são essenciais para a simulação cirúrgica realista.

Satava (1993, 1994) identificou cinco elementos que afetam o realismo em aplicações médicas para representação, visualização e manipulação de dados médicos tridimensionais (tabela 2 na página 17). Robb e Cameron (1994, 1995) ainda especificam que a taxa de atualização de vídeo deve se manter em torno de trinta *frames* por segundo e o tempo de resposta a uma ação do usuário deve ser menor que cem milisegundos (tabela 3 na página 18).

Através da análise de modelos apresentados para prover tais requisitos e para atender a tais requisitos, pode-se observar as seguintes etapas:

1. Tendo como entrada dados médicos tridimensionais capazes de prover informação morfológica (como, por exemplo, as seqüências de imagens médicas de tomografia ou ressonância magnética), aplicar um processo de segmentação sobre estes, visando identificar objetos (sólidos) distintos;

2. Com o conjunto de entrada segmentado (objetos distintos identificados dentro dos dados de entrada iniciais), gerar modelos volumétricos para representação dos objetos;

3. Aplicar aos modelos volumétricos propriedades físicas através de modelos para simulação de sólidos (considerando que os dados referentes às propriedades físicas do sólido em questão estejam disponíveis);

4. Gerar a visualização dos sólidos;

5. Através de métodos de detecção de colisão, avaliar a interação entre o usuário (através de ferramentas dependentes da aplicação) e os sólidos, atualizando os modelos volumétricos e a visualização e gerando informações sensoriais (táteis, por exemplo) quando pertinente.

Estas etapas fornecem uma visão geral do funcionamento de um aplicativo para representação, visualização e manipulação de dados médicos tridimensionais. No entanto, não são ressaltadas quais as técnicas computacionais e as estruturas de dados necessárias para obtenção dos elementos indicados por Satava e os requisitos apontados por Robb e Cameron.

A figura 28 na página 58 mostra, através de um grafo, um modelo de dependências computacionais para satisfazer os elementos de Satava. Através deste modelo, e dos outros apresentados na seção 3.1, pode-se constatar que todos os elementos dependem das estruturas de dados para representação de sólido, com excessão da fidelidade, cujos requisitos que a influenciam têm como objetivo a criação de tais estruturas.

A exposição de propriedades dos órgãos deixa clara a necessidade de uma representação de sólidos por bordas com grande informação relacional entre seus componentes, já que os métodos utilizados para obtenção deste elemento, como os métodos de simulação de sólidos apresentados na seção 2.1, realizam propagação dos fenômenos pela malha de representação do objeto. Um nível de informação relacional elevado permite maior agilidade durante a execução dos processos. Já a exposição de reações dos órgãos requer a utilização de representações de sólido por interior para detecção de atributos do objeto representado, como cavidades por exemplo. Além disso, métodos para simulação de fluidos, para simulação de sangue e bile por exemplo, exigem grande flexibilidade das estruturas de representação, pois assumem as mais variadas formas.

Apesar de satisfazer os elementos de Satava, o modelo da figura 28 pode não ser capaz de atender aos requisitos indicados por Robb e Cameron, devido a quantidade grande de dados a ser visualizada e a detecções de colisão ineficientes. Para atender a estes requisitos são introduzidas técnicas de computação gráfica em tempo real, comumente utilizadas em jogos. Estas técnicas de computação gráfica em tempo real visam eliminar do processamento gráficospartes não vistas dos objetos a partir do ponto de vista do observador, ajuste do nível de detalhes da estrutura, compatibilizando-o com as capacidades visuais do usuário e de desempenho do equipamento de execução, e indexação espacial de objetos simplificados para representação dos objetos complexos através de estruturas de dados para representação por interior, agilizando os testes de intersecção. Todas estas técnicas dependem da capacidade de se analisar e manipular as estruturas de dados para representação de sólidos. A introdução destas técnicas no modelo da figura 28 gera o modelo geral para simulação cirúrgica imersiva, mostrado na figura 33 na página 64.

Para a representação de sólidos por interior, as *octrees* são bastante utilizadas devido ao fato de serem potencialmente mais compactas e mais ágeis para operações de conjuntos do que as estruturas conhecidas como *BSP-trees*. Além disto, as *octrees* podem ser utilizadas para segmentação dos dados tridimensionais. A BONO (*Branch-on-Need Octree*) é uma variação da *octree* clássica adaptada para representação de dados providos através de imagens, adicionando novas vantagens as previamente obtidas pela *octree* original. As *octrees* podem ser facilmente utilizadas para detecção de atributos dos objetos e para a indexação do espaço para a indexação espacial, requerida pelas técnicas de detecção de colisão em tempo real.

No entanto, as *octrees* não são eficientes para a visualização e simulação realista, devido a falta de clareza imediata da localização das bordas do objeto. Para este tipo de procedimento são indicadas as representações de sólidos por bordas. A representação de sólidos por bordas representam uma malha poligonal com o formato do objeto. À esta malha, outros valores podem ser agregados, como os dados referentes às propriedades físicas do objeto. Ainda, as malhas poligonais podem ser criadas com o auxílio das *octrees* (HO-LE, 1988; WILHELMS; GELDER, 1992).

As estruturas de dados para representação de sólidos por bordas devem possuir alta infor-

mação relacional entre os componentes desta, para possibilitar a propagação de informações, e estas informações relacionais devem ser atualizadas de forma ágil durante a manipulação da malha. Várias estruturas obtêm esta característica através da representação de malhas com propriedade topológica *2-manifold* (ver seção 2.2.2) e, impondo regras de manipulação que mantêm sempre esta propriedade, conseguem manter as informações relacionais atualizadas. Ainda, é facilmente determinável em malhas com propriedade *2-manifold* quais faces estão voltadas para o lado oposto a partir do ponto de vista do observador (*culling* de faces ocultas, ver seção 3.2), pois, segundo as propriedades da DLFL, todas as faces são orientadas no mesmo sentido (AKENINE-MÖLLER; HAINES, 2002). As regras de manipulação são impostas através de operadores, utilizados para acessar a estrutura. Várias estruturas são conhecidas para manter estas propriedades, no entanto ressalta-se a DLFL (*Doubly Linked Face List*), por centralizar as informações relacionais nas faces, por onde geralmente ocorrem as interações, e por possuir apenas quatro operadores simples.

Apesar de muito eficientes no trato das informações relacionais, as estruturas de dados para representação de malhas com propriedade *2-manifold* são inflexíveis para a representação de estruturas conhecidas como *non-manifolds*, as quais podem surgir através da simulação de fluidos, os quais assumem as mais variadas formas, e das operações de conjunto sobre malhas. Além disto, a DLFL não é apta a representar *2-manifolds* com bordas e buracos nos objetos. Para estes fins, são realizadas pequenas modificações estruturais na DLFL, introduzindo novos operadores.

As expansões na DLFL, para que a abrangência de sua representação seja ampliada, ocorrem de forma que as propriedades relacionais obtidas pela estrutura original sejam mantidas. A representação de *2-manifolds* com bordas se dá através de rotulação das faces, modificando os operadores de inserção e remoção de vértices para realizar a propagação dos operadores quando necessário. Também são incluídos operadores para inserção, remoção e verificação dos rótulos. Além disso, os rótulos podem ser interpretados como os anéis da *half-edge*, os quais representam um buraco na estrutura.

Para manter as propriedades adquiridas através da topologia *2-manifold*, Mäntylä (1988)

propôs que a representação de malhas *non-manifold* seja realizada através de partes *2-manifold* e junções entre estas nos pontos *non-manifold*. As representações de malhas *non-manifold* através da DLFL, sem que se percam as informações relacionais, exigem a expansão do conceito desta estrutura. A DLFL é a implementação de um sistema de rotação de grafo, o qual foi mostrado por Edmonds (apud AKLEMAN; CHEN, 1999) equivaler sempre a uma malha *2-manifold* orientada. Um sistema de rotação de grafo é composto por uma lista de rotações, uma para cada vértice deste, sendo uma rotação de um determinado vértice uma permutação cíclica das arestas incidentes a este. Adicionando uma segunda rotação para um determinado vértice, o que transforma este em um vértice *non-manifold*, obtêm-se o mesmo efeito de uma inserção de um novo vértice com as mesmas coordenadas. Esta nova inserção representaria a mesma posição geométrica mas seria uma entidade topológica diferente. A vantagem da inserção de uma nova rotação, ao invés de uma nova inserção de vértice, é a utilização de uma única entidade topológica *non-manifold* que mantém informação relacional entre as duas partes *2-manifold* conectadas a ela. Uma aresta *non-manifold* é então representada por dois vértices *non-manifold* e uma face *non-manifold* representada por um conjunto de arestas *non-manifold*. Um operador é apresentado para a inserção de uma nova rotação em um determinado vértice, permitindo a criação de uma nova parte *2-manifold* conectada às previamente existentes.

Estes novos adendos à DLFL tornam esta uma estrutura bastante flexível, podendo ser utilizada em diversas aplicações além da medicina. O mesmo modelo apresentado pela figura 33 na página 64, pode ser, também, utilizado em diversas aplicações, pois se baseia em técnicas genéricas de sistemas gráficos, muitas vezes utilizadas em jogos e outras aplicações dependentes de desempenho. Excluindo os métodos de segmentação e geração de malha, os quais são dependentes dos dados de entrada, o modelo como um todo permanece válido para quaisquer aplicações imersivas e de simulação realista.

Mantendo a idéia de que a estrutura pode ser utilizada nas mais diversas aplicações, é apresentado um método para construção desta a partir de uma malha de entrada arbitrária. O único requisito necessário para a malha de entrada é que esta indique a organização das faces. O método de construção é capaz de identificar na malha as partes *2-manifold*, *2-manifold* com

bordas e os pontos de junção entre as partes, formando um conjunto *non-manifold*. Além disso o método orienta as faces em um sentido único em cada parte *2-manifold*. A figura 46 mostra a inserção de uma malha arbitrária, composta por duas partes *2-manifold* conectadas, em uma DLFL através do método apresentado.

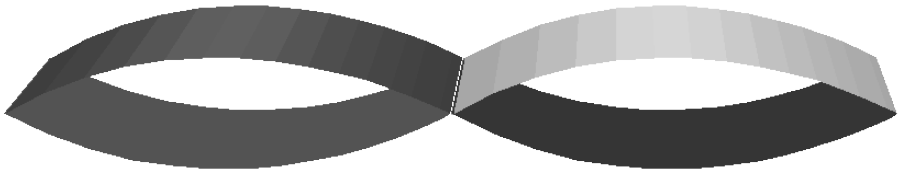

Figura 46: Inserção de uma malha, composta por duas partes *2-manifold* conectadas, em uma DLFL expandida.

Referências

AKENINE-MÖLLER, Tomas; HAINES, Eric. *Real-Time Rendering*. 2. ed. Natick, Massachussets, USA: A K Peters, 2002.

AKLEMAN, Ergun; CHEN, Jianer. Guaranteeing 2-manifold property for meshes. In: *Proceedings of Shape Modeling'99*. [S.l.: s.n.], 1999. p. 18–25.

AKLEMAN, Ergun; CHEN, Jianer. Guarateeing the 2-manifold property for meshes with doubly linked face list. *International Journal of Shape Modeling*, v. 5, n. 2, p. 149–177, 2000.

AKLEMAN, Ergun; CHEN, Jianer. *Progressive Refinement with Topological Simplification*. Tamu College Station, Texas, USA, 2003.

AKLEMAN, Ergun; CHEN, Jianer. *Topologically Robust Mesh Modeling: Concepts, Data Structures and Operations*. Tamu College Station, Texas, USA, 2003.

AKLEMAN, Ergun; CHEN, Jianer; ERYOLDAS, Fusun; SRINIVASAN, Vinod. Handle and hole improvement by using new corner cutting subdivision scheme with tension. In: *Proceedings of Shape Modeling 2001*. [S.l.: s.n.], 2001. p. 32–41.

AKLEMAN, Ergun; CHEN, Jianer; SRINIVASAN, Ergun. Interactive rind modeling. In: *Proceedings of Shape Modeling International 2003*. [S.l.: s.n.], 2003.

AKLEMAN, Ergun; CHEN, Jianer; SRINIVASAN, Vinod. A new paradigm for changing topology during subdivision modeling. In: *Proceedings of Pacific Graphics 2000*. [S.l.: s.n.], 2000. p. 192–201.

AKLEMAN, Ergun; CHEN, Jianer; SRINIVASAN, Vinod. A minimal and complete set of operators for the development of robust manifold mesh modelers. *Graphical Models*, v. 65, n. 5, p. 286–304, 2003.

AKLEMAN, Ergun; CHEN, Jianer; SRINIVASAN, Vinod; ERYOLDAS, Fusun. A new corner cutting scheme with tension and handle-face reconstruction. *International Journal of Shape Modeling*, v. 7, n. 2, p. 111–121, 2001.

AKLEMAN, Ergun; SRINIVASAN, Vinod; MANDAL, Esan. Remeshing schemes for semi-regular tilings. In: *Proceedings of Shape Modeling International 2005*. [S.l.: s.n.], 2005.

AL-KHALIFAH, Ali; ROBERTS, D. J. Survey of modelling approaches for medical simulators. In: *The International Conference Series on Disability, Virtual Reality and Associated Technologies*. [S.l.: s.n.], 2004.

AMERICAN HEART ASSOCIATION. *Computer Imaging / Tomography*. 2004. Disponível em: <http://www.americanheart.org/presenter.jhtml?identifier=4554>. Acesso em: 22/11/2004.

ARTHUR, Charles. Did reality move for you? *New Scientist*, v. 134, p. 22–27, 1991.

ARTHUR, Kevin W.; BOOTH, Kellogg S.; WARE, Colin. Evaluating 3d task performance for fish tank virtual worlds. *ACM Transactions on Information Systems*, v. 11, n. 3, p. 239–265, 1993.

AYACHE, Nicholas. Medical image analysis and simulation. In: *ASIAN '97: Proceedings of the Third Asian Computing Science Conference on Advances in Computing Science*. London, UK: Springer-Verlag, 1997. p. 4–17.

AYACHE, N.; CINQUIN, P.; COHEN, L.; LEITNER, F.; MONGA, O. Segmentation of complex 3d medical objects: A challenge and a requirement for computer assisted surgery planning and performing. In: TAYLOR, R.; LAVALLEE, S.; BURDEA, G. (Ed.). *Computer Integrated Surgery*. [S.l.]: MIT Press, 1995. p. 59–74.

BAKER, Timothy J. Adaptive modification of time evolving meshes. *Computer Methods in Applied Mechanics and Engineering*, v. 194, n. 48-49, p. 4977–5001, 2005.

BARFIELD, Woodrow; HENDRIX, Claudia. Factors affecting presence and performance in virtual environments. *Interactive Technology and the New Paradigm for Healthcare*, IOS Press, p. 21–28, 1995.

BASTOS, Alexandre C.; KORAH, Ipeson P.; CENDES, Fernando; MELANSON, Denis; TAMPIERI, Donatella; PETERS, Terry; DUBEAU, François; ANDERMANN, Frederick. Curvilinear reconstruction of 3d magnetic resonance imaging in patients with partial epilepsy: A pilot study. *Magnetic Resonance Imaging*, v. 13, n. 8, p. 1107–1112, 1995.

BAUMGART, Bruce G. *Winged Edge Polyhedron Representation*. Stanford, CA, USA, 1972. CS-TR-72-320.

BAUMGART, Bruce G. *Geometric Modelling for Computer Vision*. Tese (Doutorado) — Stanford University, 1974.

BAUMGART, Bruce G. A polyhedron representation for computer vision. In: *National Computer Conference*. [S.l.: s.n.], 1975. p. 589–596. AFIPS Conference Proceedings.

BERGEN, Gino van den. *Collision Detection in Interactive 3D Environments*. [S.l.]: Morgan Kaufmann, 2003.

BHATTACHARYA, Parthajit. *Efficient Neighbor Finding Algorithms in Quadtree and Octree*. Dissertação (Mestrado) — Department of Computer Science & Engineering, Indian Institute of Technology, Kanpur, India, 2001.

BOADA, Imma; NAVAZO, Isabel. An octree isosurface codification based on discrete planes. In: *Proceedings of the 17th Spring conference on Computer graphics*. [S.l.: s.n.], 2001. p. 130–137.

BORODIN, Pavel; ZACHMANN, Gabriel; KLEIN, Reinhard. Consistent normal orientation for polygonal meshes. In: *Proc. Computer Graphics International 2004 (CGI'04)*. [S.l.: s.n.], 2004. p. 18–25.

BOURG, David. *Physics for Game Developers*. [S.l.]: O'Reilly Media, 2001.

BURG, John van der. Building an advanced particle system. *Game Developer Magazine*, p. 44–50, March 2000.

ÇAKMAK, Hüseyin; KÜHNAPFEL, Uwe. Animation and simulation techniques for vr-training systems in endoscopic surgery. In: *Eurographics Workshop on Animation and Simulation '2000 (EGCAS '2000)*. [S.l.: s.n.], 2000. p. 172–185.

ÇAKMAK, Hüseyin. K.; KÜHNAPFEL, Uwe; BRETTHAUER, Georg. Virtual reality techniques for education and training in minimally invasive surgery. In: *Proceedings of VDE World Micro Technologies Conference*. [S.l.: s.n.], 2000. p. 395–400.

CATMULL, Edwin; CLARK, Jim. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, v. 10, n. 6, p. 350–355, 1978.

CHADWICK, John E.; HAUMANN, David R.; PARENT, Richard E. Layered construction for deformable animated characters. *SIGGRAPH Computer Graphics*, v. 23, n. 3, p. 243–252, 1989.

CHEN, Jianer. Algorithmic graph embeddings. *Theoretical Computer Science*, v. 181, n. 2, p. 247–266, 1997.

CHRISTESSEN, A. H. J. Approximation of a donut. In: *Computer Graphics*. [S.l.: s.n.], 1980. v. 14, n. 3. Front page picture of Proc. SIGGRAPH'80.

DACHILLE, IX Frank; QIN, Hong; KAUFMAN, Arie; EL-SANA, Jihad. Haptic sculpting of dynamic surfaces. In: *SI3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics*. New York, NY, USA: ACM Press, 1999. p. 103–110.

DELINGETTE, Hervé. Towards realistic soft tissue modeling in medical simulation. In: *Proceedings of the IEEE: Special Issue on Surgery Simulation*. [S.l.: s.n.], 1998. p. 512–523.

DESBRUN, Mathieu; SCHRÖDER, Peter; BARR, Alan. Interactive animation of structured deformable objects. In: *Proceedings of the 1999 conference on Graphics interface '99*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999. p. 1–8.

DHATT, Gouri; TOUZOT, Gilbert. *The Finite Element Method Displayed*. New York, NY, USA: Wiley, 1984.

DIONISIO, Jose; HENRICH, Volker; JAKOB, Udo; RETTIG, Alexander; ZIEGLER, Rolf. The virtual touch: Haptic interfaces in virtual environments. *Computers & Graphics*, v. 21, n. 4, p. 459–468, 1997.

DOO, Daniel; SABIN, Malcolm. Analysis of the behaviour of recursive division surfaces near extraordinary points. *Computer Aided Design*, v. 10, n. 6, p. 356–360, 1978.

EBERLY, David H. *3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics*. [S.l.]: Morgan Kaufmann, 2000.

EBERLY, David H. *Game Physics*. [S.l.]: Morgan Kaufmann, 2003.

EBERLY, David H. *3D Game Engine Architecture: Engineering Real-Time Applications with Wild Magic*. [S.l.]: Morgan Kaufmann, 2004.

EDMONDS, Jack. A combinatorial representation for polyhedral surfaces. *Notices American Mathematical Society*, v. 7, p. 646, 1960.

EPPSTEIN, David; GOODRICH, Micheal T.; SUN, Jonathan Zheng. The skip quadtree: A simple dynamic data structure for multidimensional data. In: *Proc. 21th Symposium on Computational Geometry*. [S.l.]: ACM Press, 2005. p. 296–305.

FLORIANI, Leila De; HUI, Annie. A scalable data structure for three-dimensional non-manifold objects. In: *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2003. p. 72–82.

FOLEY, James D.; DAM, Andries van; FEINER, Steven K.; HUGHES, John H. *Computer Graphics - Principles and Practice*. 2. ed. Reading, Massachusetts, USA: Addison-Wesley, 1990.

FOURNIER, Alain; FUSSELL, Don; CARPENTER, Loren. Computer rendering of stochastic models. *Communications of the ACM*, v. 25, n. 6, p. 371–384, 1982.

FREY, Pascal J. Generation and adaptation of computational surface meshes from discrete anatomical data. *International Journal for Numerical Methods in Engineering*, v. 60, p. 1049–1074, 2004.

FRISKEN, Sarah F.; PERRY, Ronald N. Simple and efficient traversal methods for quadtrees and octrees. *Journal of Graphics Tools*, v. 7, n. 3, p. 1–11, 2002.

FUCHS, Henry; KEDEM, Zvi M.; NAYLOR, Bruce R. On visible surface generation by a priori tree structures. In: *Proceedings of the 7th annual conference on Computer graphics and interactive techniques*. [S.l.: s.n.], 1980. p. 124–133.

GARGANTINI, Irene. Linear octree for fast processing of three dimensional objects. *Computer Graphics and Image Processing*, v. 20, n. 4, p. 356–374, 1982.

GEORGE, P. L.; BOROUCHAKI, H.; LAUG, P. An efficient algorithm for 3d adaptive meshing. *Advances in Engineering Software*, v. 33, n. 7-10, p. 377–387, 2002.

GEORGII, Joachim; WESTERMANN, Rüdiger. Interactive simulation and rendering of heterogeneous deformable bodies. In: *Vision, Modeling and Visualization 2005*. [S.l.: s.n.], 2005.

GIBSON, Sarah; FYOCK, Christina; GRIMSON, Eric; KANADE, Takeo; KIKINIS, Ron; LAUER, Hugh; MCKENZIE, Neil; MOR, Andrew; NAKAJIMA, Shin; OHKAMI, Hide; OSBORNE, Randy; SAMOSKY, Joseph; SAWADA, Akira. Simulating surgery using volumetric object representations, real-time volume rendering and haptic feedback. *Medical Image Analysis*, n. 121–132, 1998.

GIBSON, Sarah F. 3d chainmail: a fast algorithm for deforming volumetric objects. In: *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*. New York, NY, USA: ACM Press, 1997. p. 149–154.

GIBSON, Sarah F. F.; MIRTICH, Brian. *A Survey of Deformable Modeling in Computer Graphics*. Cambridge, MA, USA, 1997. TR97-19.

GIBSON, Sarah F. Frisken; SAMOSKY, Joe; MOR, Andrew; FYOCK, Christina; GRIMSON, W. Eric L.; KANADE, Takeo; KIKINIS, Ron; LAUER, Hugh C.; MCKENZIE, Neil; NAKAJIMA, Shin; OHKAMI, TakaHide; OSBORNE, Randy; SAWADA, Akira. Simulating arthroscopic knee surgery using volumetric object representations, real-time volume rendering and haptic feedback. In: *CVRMed-MRCAS '97: Proceedings of the First Joint Conference on Computer Vision, Virtual Reality and Robotics in Medicine and Medial Robotics and Computer-Assisted Surgery*. London, United Kingdom: Springer-Verlag, 1997. p. 369–378.

GUEORGUIEVA, Stefka; MARCHEIX, David. Non-manifold boundary representation for solid modeling. In: *Proceedings of the International Computer Symposium*. [S.l.: s.n.], 1994.

GUÉZIEC, Andre; HUMMEL, Robert. The wrapper algorithm: Surface extraction and simplificationthe wrapper algorithm: Surface extraction and simplification. In: *Proc. IEEE Workshop on Biomedical Image Analysis*. [S.l.: s.n.], 1994. p. 204–213.

GUIBAS, Leonidas J.; STOLFI, Jorge. Primitives for the manipulation of general subdivisions and the computation of voronoi diagrams. *ACM Transactions on Graphics*, v. 4, n. 2, p. 74–123, 1985.

HAHN, James K. Realistic animation of rigid bodies. In: *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM Press, 1988. p. 299–308.

HEARN, Donald; BAKER, M. Pauline. *Computer Graphics - C Version*. 2. ed. [S.l.]: Prentice Hall, 1996.

HEFFTER, L. Uber das problem der nachbargebiete. *Math. Annalen*, v. 38, p. 477–508, 1891.

HIGASHI, Masatake; TORIHARA, Fuyuki; TAKEUCHI, Nobuhiro; SATA, Toshio; SAITOH, Tsuyoshi; HOSAKA, Mamoru. Face-based data structure and its application to robust geometric modeling. In: *SMA '95: Proceedings of the third ACM symposium on Solid modeling and applications*. [S.l.: s.n.], 1995. p. 235–246.

HIGASHI, Masatake; YATOMI, Hideki; MIZUTANI, Yoshihiro; MURABATA, Shin ichi. Unified geometric modeling by non-manifold shell operation. In: *SMA '93: Proceedings on the second ACM symposium on Solid modeling and applications*. New York, NY, USA: ACM Press, 1993. p. 75–84.

HO-LE, K. Finite element mesh generation methods: A review and classification. *Computer Aided Design*, v. 20, n. 1, p. 27–38, 1988.

HOFFMANN, Christoph M.; HOPCROFT, John E. Geometric ambiguities in boundary representations. *Computer Aided Design*, v. 19, n. 3, p. 141–147, 1987.

HOFFMANN, Christoph M.; VANECEK, G. Fundamental techniques for geometric and solid modeling. *Manufacturing and Automation Systems: Techniques and Technologies*, n. 48, p. 101–165, 1990.

HOLLOWAY, Richard; FUCHS, Henry; ROBINETT, Warren. Virtual-worlds research at the university of north carolina at chapel hill as of february 1992. In: *CG International '92: Proceedings of the 10th International Conference of the Computer Graphics Society on Visual computing : integrating computer graphics with computer vision*. New York, NY, USA: Springer-Verlag New York, Inc., 1992. p. 109–128.

HOROWITZ, Steven L.; PAVLIDIS, Theodosios. Picture segmentation by a tree traversal algorithm. *Journal of the ACM (JACM)*, v. 23, n. 2, p. 368–388, 1976.

HUANG, Weizhang. Variational mesh adaptation: Isotropy and equidistribution. *Journal of Computational Physics*, v. 174, p. 903–924, 2001.

HUNTER, Michael G. *Efficient computation and data structures for graphics*. Tese (Doutorado) — Department of Electrical Engineering and Computer Science, Princeton University, Princeton, NJ, USA, 1978.

JACKINS, Chris L.; TANIMOTO, Steven L. Oct-trees and their use in representing three-dimensional objects. *Computer Grpahics and Image Processing*, v. 14, n. 3, p. 249–270, 1980.

JAILLET, Fabrice; SHARIAT, Behzad; VANDORPE, Denis. Deformable volume object modeling with a particle-based system for medical applications. In: *Proc. Computer Graphics and Visualization WSCG'97*. [S.l.: s.n.], 1997. p. 192–201.

KALLINDERIS, Yannis; KAVOUKLIS, Christos. A dynamic adaptation scheme for general 3-d hybrid meshes. *Computer Methods in Applied Mechanics and Engineering*, v. 194, n. 48-49, p. 5019–5050, 2005.

KALTENBORN, K. F.; RIENHOFF, O. Virtual reality in medicine. *Methods of Information in Medicine*, v. 32, n. 3, p. 407–417, 1993.

KANAI, Takashi. Meshtoss: Converting subdivision surfaces from dense meshes. In: *Proc. 6th International Workshop on Vision, Modeling and Visualization 2001*. Amsterdam: IOS Press, 2001. p. 325–332.

KOBBELT, Leif P.; BOTSCH, Mario; SCHWANECKE, Ulrich; SEIDEL, Hans-Peter. Feature sensitive surface extraction from volume data. In: *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM Press, 2001. p. 57–66.

KRYSL, Petr; ORTIZ, Michael. Extraction of boundaru representation from surface triangulations. *International Journal for Numerical Methods in Engineering*, v. 50, p. 1737–1758, 2001.

KÜHNAPFEL, Uwe; ÇAKMAK, Hüseyin K.; MAASS, Heiko; WALDHAUSEN, S. *Models for simulating instrument-tissue interactions*. 2001. Paper presented at Medicine Meets Virtual Reality.

LABSIK, Ulf; HORMANN, Kai; MEISTER, Martin; GREINER, Günther. Hierarchical iso-surface extraction. *Journal of Computing and Information Science in Engineering*, v. 2, n. 4, p. 323–329, 2002.

LAKARE, Sarang. *3D Segmentation Techniques for Medical Volumes*. Stony Brook, NY, USA, 2000. Research Prociency Exam.

LAMOTHE, André. *Tricks of the 3D Game Programming Gurus: Advanced 3D Graphics and Rasterization*. [S.l.]: Sams, 2003.

LANDAU, Lev; LIFSHITZ, E. *Theory of Elasticity*. 3. ed. [S.l.]: Butterworth-Heinemann, 1986.

LARSSON, Thomas; AKENINE-MÖLLER, Tomas. A dynamic bounding volume hierarchy for generalized collision detection. In: *Proc. Workshop On Virtual Reality Interaction and Physical Simulation*. [S.l.: s.n.], 2005. p. 91–100.

LEE, John M. *Introduction to Topological Manifolds*. [S.l.]: Springer, 2000.

LEE, Sang Hun; LEE, Kunwoo. Partial entity structure: a compact non-manifold boundary representation based on partial topological entities. In: *SMA '01: Proceedings of the sixth ACM symposium on Solid modeling and applications*. New York, NY, USA: ACM Press, 2001. p. 159–170.

LEIGH, Jason; JOHNSON, Andrew E.; BROWN, Maxine; SANDIN, Daniel J.; DEFANTI, Thomas A. Visualization in teleimmersive environments. *IEEE Computer*, v. 32, n. 12, p. 66–73, 1999.

LENGYEL, Eric. *Mathematics for 3D Game Programming & Computer Graphics*. 1. ed. [S.l.]: Charles River Media, 2001.

LEVOY, Marc. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, v. 8, n. 3, p. 29–37, 1988.

LEVOY, Marc. Efficient ray tracing of volume data. *ACM Transactions on Graphics*, v. 9, n. 3, p. 245–261, 1990.

LEWINER, Thomas; LOPES, Hélio; VIEIRA, Antônio Wílson; TAVARES, Geovan. Efficient implementation of marching cubes' cases with topological guarantees. *Journal of Graphics Tools*, v. 8, n. 2, p. 1–15, 2003.

LIBES, Don. Modeling dynamic surfaces with octrees. *Computer & Graphics Magazine*, v. 15, p. 383–387, 1991.

LIN, Ming C.; OTADUY, Miguel A. Sensation-preserving haptic rendering. *IEEE Computer Graphics and Applications*, v. 25, p. 8–11, 2005.

LOK, Benjamin C. Toward the merging of real and virtual spaces. *Communications of the ACM*, v. 47, n. 8, p. 48–53, 2004.

LOMBARDO, Jean-Christophe; CANI, Marie-Paule; NEYRET, Fabrice. Jean-christophe lombardo and marie-paule cani and fabrice neyret. In: *CA '99: Proceedings of the Computer Animation*. Washington, DC, USA: IEEE Computer Society, 1999. p. 82–91.

LOOP, Charles. *Smooth subdivision surfaces based on triangles*. Dissertação (Mestrado) — University of Utah, Department of Mathematics, USA, 1987.

LOPES, Adriano; BRODIE, Ken. Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing. *IEEE Transactions on Visualization and Computer Graphics*, v. 9, n. 1, p. 16–29, 2003.

LORENSEN, William E.; CLINE, Harvey E. Marching cubes: A high resolution 3d surface construction algorithm. In: *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. [S.l.]: ACM Press, 1987. p. 163–169.

LUEBKE, David; REDDY, Martin; COHEN, Jonathan D.; VARSHNEY, Amitabb; WATSON, Benjamin; HUEBNER, Robert. *Level of Detail for 3D Graphics*. [S.l.]: Morgan Kaufmann, 2002.

LUO, Yi; LUKÁCS, Gábor. A boundary representation for form features and non-manifold solid objects. In: *SMA '91: Proceedings of the first ACM symposium on Solid modeling foundations and CAD/CAM applications*. New York, NY, USA: ACM Press, 1991. p. 45–60.

MA, Weiyin; MA, Xiaohu; TSO, Shiu Kit; PAN, Zhigeng. Subdivision surface fitting from a dense triangle mesh. In: *Proc. Geometric Modeling and Processing (GMP 2002), Theory and Applications*. [S.l.]: IEEE Computer Society, 2002. p. 94–103.

MACIEL, Anderson; NEDEL, Luciana Porcher; FREITAS, Carla M. Dal Sasso. Anatomy-based joint models for virtual humans skeletons. In: *Computer Animation 2002*. [S.l.: s.n.], 2002. p. 220–233.

MÄNTYLÄ, Martti. Boolean operations of 2-manifolds through vertex neighborhood classification. *ACM Transaction on Graphics*, v. 5, n. 1, p. 1–29, 1986.

MÄNTYLÄ, Martti. *Introduction to Solid Modeling*. Rockville, Maryland, USA: Computer Science Press, 1988.

MÄNTYLÄ, Martti; SULONEN, Reijo. Gwb - a solid modeler with euler operators. *IEEE Computer Graphics & Applications*, v. 2, n. 7, p. 17–31, 1982.

MÄNTYLÄ, Martti; TAKALA, Tapio. The geometric workbench (gwb) - an experimental geometric modeling system. In: *Proc. EUROGRAPHICS'81*. Amsterdam: North-Holland Publisher, 1981. p. 205–215.

MARCH, Lionel; STEADMAN, Philip. *The Geometry of Environment*. Cambridge, Massachussets, USA: MIT Press, 1971.

MARK, William R.; RANDOLPH, Scott C.; FINCH, Mark; VERTH, James M. Van; TAYLOR, II Russell M. Adding force feedback to graphics systems: issues and solutions. In: *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM Press, 1996. p. 447–452.

MATVEYEV, Sergey V. Approximation of isosurface in the marching cube: Ambiguity problem. In: *Proceedings IEEE Visualization '94*. [S.l.]: IEEE Computer Society, 1994. p. 288–292.

MCINERNEY, Tim; TERZOPOULOS, Demetri. A finite element model for 3d shape reconstruction and nonrigid motion tracking. In: *Proc. Fourth IEEE International Conference on Computer Vision*. [S.l.: s.n.], 1993. p. 518–523.

MCRAE, Scott D. Adaptive mesh algorithms - a review of progress and future research needs. In: *Proc. AIAA Computational Fluid Dynamics Conference*. [S.l.: s.n.], 2001. AIAA-2001-2551.

MEAGHER, Donald J. *Octree Encoding: A new Technique for the Representation, Manipulation, and Display of Arbitrary 3-D Objects by Computer*. Troy, NY, USA, 1980.

MILLER, Gavin S. P. The motion dynamics of snakes and worms. In: *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM Press, 1988. p. 169–173.

MOLINE, Judi A. Virtual reality for health care: A survey. *Journal of Research of the National Institute of Standards and Technology*, v. 1, n. 44, p. 3–34, 1997.

MÖLLER, Tomas; HAINES, Eric. *Real-Time Rendering*. Natick, Massachusetts, USA: A K Peters, 1999.

MOORE, Matthew; WILHELMS, Jane. Collision detection and response for computer animation. In: *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM Press, 1988. p. 289–298.

MÜLLER, Matthias; CHARYPAR, David; GROSS, Markus. Particle-based fluid simulation for interactive applications. In: *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Aire-la-Ville, Switzerland: Eurographics Association, 2003. p. 154–159.

MURALI, T. M.; FUNKHOUSER, Thomas A. Consistent solid and boundary representations from arbitrary polygonal data. In: *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*. New York, NY, USA: ACM Press, 1997. p. 155–162.

MUSSE, Olivier; ARMSPACH, Jean-Paul; NAMER, Izzie Jacques; HEITZ, Fabrice; HENNEL, Franciszek; GRUCKER, Daniel. Data-driven curvilinear reconstructions of 3d mr images: Application to cryptogenic extratemporal epilepsy. *Magnetic Resonance Imaging*, v. 16, n. 10, p. 1227–1235, 1998.

NAKAJIMA, Shin; KIKINIS, Ron; JOLESZ, Ferenc A.; ATSUMI, Hideki; LEVENTON, Micheal E.; GRIMSON, W. Eric L.; HATA, Nobuhiko; METCALF, David C.; MORIARTY, Thomas M.; BLACK, Peter McL.; GARADA, Basem; ALEXANDER, Eben III. 3d mri reconstruction for surgical planning and guidance. In: ALEXANDER, Eben III; MACIUNAS, R. J. (Ed.). *Advanced Neurosurgical Navigation*. [S.l.]: Thieme Medical Publishers, 1999. p. 137–145.

NEALAN, Andrew; MÜLLER, Matthias; KEISER, Richard; BOXERMANN, Eddy; CARLSON, Mark. Physically based deformable models in computer graphics. In: CHRYSANTHOU, Yiorgos; MAGNOR, Marcus (Ed.). *STAR Proceedings of Eurographics 2005*. Geneva, Switzerland: Eurographics Association, 2005. p. 71–94.

NIELSON, Gregory M. On marching cubes. *IEEE Transactions on Visualization and Computer Graphics*, v. 9, n. 3, p. 283–297, 2003.

NIELSON, Gregory M.; HAMANN, Bernd. The asymptotic decider: resolving the ambiguity in marching cubes. In: *VIS '91: Proceedings of the 2nd conference on Visualization '91*. Los Alamitos, CA, USA: IEEE Computer Society Press, 1991. p. 83–91.

O'ROURKE, Joseph. *Computational Geometry in C*. 2. ed. [S.l.]: Cambridge University Press, 2001.

POZRIKIDIS, C. *Fluid Dynamics: Theory, Computation, and Numerical Simulation*. [S.l.]: Springer-Verlag New York, 2001.

PRATT, David R.; ZYDA, Michael; KELLEHER, Kristen. Virtual reality: In the mind of the beholder. *IEEE Computer*, v. 28, n. 7, p. 17–19, 1995.

PROVOT, Xavier. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In: *Proc. Graphics Interface '95*. [S.l.: s.n.], 1995. p. 147–154.

REDDY, Raj; RUBIN, Steven M. *Representation of Three-Dimensional Objects*. Pittsburgh, PA, USA, 1978.

REEVES, William T. Particle systems - a technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics*, v. 2, n. 2, p. 91–108, 1983.

REEVES, William T. Particle systems - a technique for modeling a class of fuzzy objects. In: *SIGGRAPH '83: Proceedings of the 10th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM Press, 1983. p. 359–375.

REEVES, William T.; BLAU, Ricki. Approximate and probabilistic algorithms for shading and rendering structured particle systems. In: *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM Press, 1985. p. 313–322.

REQUICHA, Aristides G. Representations for rigid solids: Theory, methods, and systems. *ACM Computing Surveys*, v. 12, n. 4, p. 437–464, 1980.

RITTER, Jack. An efficient bounding sphere. In: GLASSNER, Andrew S. (Ed.). *Graphics Gems*. [S.l.]: Morgan Kaufmann, 1990. cap. 5.2, p. 301–303.

ROBB, Richard A.; CAMERON, Bruce. Virtual reality assisted surgery program. In: SATAVA, R. M.; MORGAN, K.; SIEBURG, H. B.; MATTHEUS, R.; CHRISTENSEN, J. P. (Ed.). *Interactive Technology and the New Paradigm for Healthcare*. Amsterdam: IOS Press, 1995.

ROBB, Richard A.; CAMERON, Bruce M. *VRASP: Virtual Reality Assisted Surgery Program*. 1994. Paper presented at the First International Symposium on Computer Aided Surgery.

ROSEN, Joseph. The role of telemedicine and telepresence in reducing health care costs. In: *Medicine Meets Virtual Reality II: Interactive Technology & Healthcare: Visionary Applications for Simulation Visualization Robotics*. [S.l.]: Aligned Management Associates, 1994. p. 187–194.

SALEMBIER, Philippe; GARRIDO, Luis. Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. *IEEE Transactions on Image Processing*, v. 9, n. 4, p. 561–576, 2000.

SAMET, Hanan. The quadtree and related hierarchical data structures. *ACM Computing Surveys (CSUR)*, ACM Press, v. 16, n. 2, p. 187–260, 1984.

SAMET, Hanan. *The Design and Analysis of Spatial Data Structures*. Reading, Massachusetts, USA: Addison-Wesley, 1989.

SAMET, Hanan. Neighbor finding in images represented by octrees. *Computer Vision, Graphics, and Image Processing*, v. 46, n. 3, p. 367–386, 1989.

SAMET, Hanan. *Applications of Spatial Data Structures: Computer Graphics, Image Processing and GIS*. Reading, MA, USA: Addison-Wesley, 1990.

SANTOS, Thiago Ramos dos; ABDALA, Daniel Duarte; WANGENHEIM, Aldo von. Three-dimensional visualization of radiological images using octrees. In: *Proceedings of the 17th IEEE Symposium on Computer-Based Medical Systems (CBMS'04)*. [S.l.: s.n.], 2004. p. 44–47.

SATAVA, Richard M. Robotics, telepresence, and virtual reality: A critical analysis of the future of surgery. *Minimally Invasive Therapy and Allied Technologies*, v. 1, n. 6, p. 357–363, 1992.

SATAVA, Richard M. Virtual reality surgical simulator: The first steps. In: *Proceedings of the Third Annual Conference on Virtual Reality*. [S.l.: s.n.], 1993. p. 103–105.

SATAVA, Richard M. *Medicine 2001: The King Is Dead*. 1994. Disponível em: <http://www.csun.edu/cod/conf/1994/proceedings/Med 1.htm>. Acesso em: 25/11/2004.

SCHARVER, Chris; EVENHOUSE, Ray; JOHNSON, Andrew; LEIGH, Jason. Designing cranial implants in a haptic augmented reality environment. *Communications of the ACM*, v. 47, n. 8, p. 33–38, 2004.

SCHARVER, Chris; EVENHOUSE, Ray; JOHNSON, Andrew; LEIGH, Jason. Pre-surgical cranial implant design using the paris prototype. In: *Proceedings of the IEEE Conference on Virtual Reality*. [S.l.: s.n.], 2004. p. 199–291.

SCHNEIDER, Philip; EBERLY, David H. *Geometric Tools for Computer Graphics*. [S.l.]: Morgan Kaufmann, 2002.

SCHROEDER, William J.; ZARGE, Jonathan A.; LORENSEN, William E. Decimation of triangle meshes. In: *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM Press, 1992. p. 65–70.

SHAGAM, Joshua; JR., Joseph J. Pfeiffer. *Dynamic Irregular Octrees*. Las Cruces, New Mexico, USA, 2003. NMSU-CS-2003-004.

SHAGAM, Joshua; JR., Joseph J. Pfeiffer. *Dynamic Spatial Partitioning for Real-Time Visibility Determination*. Las Cruces, New Mexico, USA, 2003. NMSU-CS-2003-006.

SHEFFER, Alla; ÜNGÖR, Alper. Efficient adaptive meshing of parametric models. In: *SMA '01: Proceedings of the sixth ACM symposium on Solid modeling and applications*. New York, NY, USA: ACM Press, 2001. p. 59–70.

SHERIDAN, Thomas B. *Telerobotics, automation, and human supervisory control*. Cambridge, Massachusetts, USA: MIT Press, 1992.

SOLLENBERGER, R. L.; MILGRAN, P. A. A comparative study of rotational and stereoscopic computer graphic depth cues. In: HUMAN FACTORS SOCIETY. *Proceeding of the Human Factors Society 35th Annual Meeting*. [S.l.], 1991. p. 1452–1456.

SRIHARI, Sargur N. Representation of three-dimensional digital images. *ACM Computing Surveys*, v. 13, n. 4, p. 399–424, 1981.

SRINIVASAN, Vinod. *Modeling High-Genus Surfaces*. Tese (Doutorado) — Texas A&M University, Tamu College Station, Texas, USA, 2004.

SRINIVASAN, Vinod; AKLEMAN, Ergun; CHEN, Jianer. Interactive construction of multi-segment curved handles. In: *Proceedings of Pacific Graphics 2002*. [S.l.: s.n.], 2002.

SRINIVASAN, Vinod; AKLEMAN, Ergun; KEYSER, John. *Topological Construction of 2-Manifold Meshes from Arbitrary Polygonal Data*. Tamu College Station, Texas, USA, 2004.

STAM, Jos; LOOP, Charles. Quad/triangle subdivision. *Computer Graphics Forum*, v. 22, n. 1, p. 79–86, 2003.

SZELISKI, Richard; TONNESEN, David. Surface modeling with oriented particle systems. In: *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM Press, 1992. p. 185–194.

TERZOPOULOS, Demetri; PLATT, John; BARR, Alan; FLEISCHER, Kurt. Elastically deformable models. In: *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM Press, 1987. p. 205–214.

TESCHNER, M.; KIMMERLE, S.; HEIDELBERGER, B.; ZACHMANN, G.; RAGHUPATHI, Laks; FUHRMANN, A.; CANI, Marie-Paule; FAURE, François; MAGNETAT-THALMANN, N.; STRASSER, W.; VOLINO, P. Collision detection for deformable objects. *Computer Graphics Forum*, v. 24, n. 1, p. 61–81, 2005.

THIBAULT, William C.; NAYLOR, Bruce F. Set operations on polyhedra using binary space partitioning trees. In: *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. [S.l.: s.n.], 1987. p. 153–162.

TRISTANO, Joseph R.; CHEN, Zhijan; HANCQ, D. Alfred; KWOK, Wa. Fully automatic adaptive mesh refinement integrated into the solution process. In: *Proc. 12th International Meshing Roundtable*. [S.l.: s.n.], 2003. p. 307–314.

WAGNER, Harley M. *Atlas Cerebral Digital: Desenvolvimento de uma Ferramenta Computacional para Mapeamento Funcional e Anatômico de Áreas Cerebrais, Baseado no Atlas de Talairach*. Dissertação (Mestrado) — Departamento de Informática e Estatística, Universidade Federal de Santa Catarina, Florianópolis, Santa Catarina, Brazil, 2001.

WARREN, Joe D.; SCHAEFER, Scott. A factored approach to subdivision surfaces. *IEEE Computer Graphics and Applications*, v. 24, n. 3, p. 74–81, 2004.

WATT, Alan; WATT, Mark. *Advanced Animation and Rendering Techniques: Theory and Practice*. [S.l.]: Addison-Wesley Professional, 1992.

WEILER, Kevin. Edge-based data structures for solid modeling in curved-surface environments. *IEEE Computer Graphics & Applications*, v. 5, n. 1, p. 21–40, 1985.

WEINGÄRTNER, Tim; DILLMANN, Rüdiger. *Split-and-Merge Segmentation using Octrees*. 1995. Disponível em: <citeseer.ist.psu.edu/223834.html>.

WILHELMS, Jane; GELDER, Allen van. Octrees for faster isosurface generation. *ACM Transactions on Graphics*, v. 11, n. 3, p. 201–227, 1992.

WU, Yi; THALMANN, Daniel; THALMANN, Nadia Magnenat. Deformable surfaces using physically-based particle systems. In: *Proc. Computer Graphics International'05*. [S.l.]: Academic Press, 1995. p. 205–216.

YAMAGUSHI, Yasushi; KIMURA, Fumihiko. Nonmanifold topology based on coupling entities. *IEEE Computer Graphics and Applications*, v. 15, n. 1, p. 42–50, 1995.

YAU, Mann-May; SRIHARI, Sargur N. A hierarchical data structure for multidimensional digital images. *Communications of the ACM*, ACM Press, v. 26, n. 7, p. 504–515, 1983.

YING, Lexing; ZORIN, Denis. Nonmanifold subdivision. In: *VIS '01: Proceedings of the conference on Visualization '01*. Washington, DC, USA: IEEE Computer Society, 2001. p. 325–332.

YOO, Terry S.; RHEINGANS, Penny. Digital design of a surgical simulator for interventional mr imaging (case study). In: *VIS '99: Proceedings of the conference on Visualization '99*. Los Alamitos, CA, USA: IEEE Computer Society Press, 1999. p. 393–396.

ZERFASS, Peter; KEEVE, Erwin. Towards a virtual environment for biomechanical simulation. In: *CARS 2001. Computer Assisted Radiology and Surgery. Proceedings of the 15th International Congress and Exhibition*. [S.l.]: Elsevier, 2001. p. 73–78.

ZHANG, Yongjie; BAJAJ, Chandrajit; SOHN, Bong-Soo. 3d finite element meshing from imaging data. *Computer Methods in Applied Mechanics and Engineering*, v. 194, n. 48-49, p. 5083–5106, 2005.

ZIENKIEWICZ, Olgierd C. *The Finite Element Method*. 3. ed. London, UK: McGraw-Hill, 1977.

ZIENKIEWICZ, Olgierd C. *The Finite Element Method*. 5. ed. [S.l.]: Butterworth-Heinemann, 2000.

ZORIN, Denis; SCHRÖDER, Peter. *Subdivision for Modeling and Animation*. 2000. ACM SIGGRAPH'2000 Course Notes n. 23. Editors.

ZORIN, Denis; SCHRÖDER, Peter; SWELDENS, Wim. Interpolating subdivision for meshes with arbitrary topology. In: *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM Press, 1996. p. 189–192.