UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
FACULDADE DE ENGENHARIA

PAULO JÓIA FILHO

RECONSTRUÇÃO E GERAÇÃO DE MALHAS EM ESTRUTURAS BIOMECÂNICAS TRIDIMENSIONAIS PARA ANÁLISE POR ELEMENTOS FINITOS

Bauru - SP

2008

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
FACULDADE DE ENGENHARIA

PAULO JÓIA FILHO

RECONSTRUÇÃO E GERAÇÃO DE MALHAS EM ESTRUTURAS BIOMECÂNICAS TRIDIMENSIONAIS PARA ANÁLISE POR ELEMENTOS FINITOS

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Mecânica, para obtenção do grau de Mestre em Engenharia Mecânica.

Orientador: Prof. Dr. Edson A Capello Sousa

Bauru - SP

2008

DIVISÃO TÉCNICA DE BIBLIOTECA E DOCUMENTAÇÃO
UNESP – BAURU

Jóia Filho, Paulo.
Reconstrução e geração de malhas em estruturas biomecânicas tridimensionais para análise por elementos finitos / Paulo Jóia Filho, 2008.
xvi, 118 f. il.

Orientador: Edson Antônio Capello Sousa.

Dissertação (Mestrado)– Universidade Estadual Paulista. Faculdade de Engenharia, Bauru, 2008.

1. Biomecânica. 2. Método dos elementos finitos. 3. Imagem (Medicina). 4. Reconstrução tridimensional. 5. Geração de malha. I. Universidade Estadual Paulista. Faculdade de Engenharia. II. Título.

Ficha catalográfica elaborada por Maricy Fávaro Braga – CRB-8 1.622

PAULO JÓIA FILHO

RECONSTRUÇÃO E GERAÇÃO DE MALHAS EM ESTRUTURAS BIOMECÂNICAS TRIDIMENSIONAIS PARA ANÁLISE POR ELEMENTOS FINITOS

Dissertação submetida à Comissão Examinadora designada pelo Colegiado do Curso de Pós-Graduação em Engenharia Mecânica da Universidade Estadual Paulista "Júlio de Mesquita Filho" Faculdade de Engenharia como requisito para obtenção do grau de Mestre em Engenharia Mecânica.

Bauru, 27 de junho de 2008.

BANCA EXAMINADORA

_____

Prof. Dr. Edson Antônio Capello Sousa

FEB / UNESP – Bauru

_____

Prof. Adjunto Sérgio Scheer

Centro Politécnico – UFPR

_____

Prof. Dr. José Eduardo Cogo Castanho

FEB / UNESP – Bauru

À minha esposa Andréia,
que pela confiança e apoio sempre
me impulsionou a seguir adiante.

AGRADECIMENTOS

Agradeço a todos aqueles que contribuíram para o desenvolvimento deste trabalho.

Em especial à minha amada esposa Andréia Chudrik Jóia, pela paciência e cooperação incessante nesta árdua tarefa.

Ao Prof. Dr. Edson A. Capello Sousa, pela ajuda e orientação do trabalho.

Aos professores do mestrado em Engenharia Mecânica, pelos novos ensinamentos.

Ao Prof. Dr. Yukio Kobayashi pelas nossas conversas casuais e amigas, onde sempre demonstrou incentivo e apoio.

À Supervisora da Seção de Pós-graduação, Célia Cristina, por sua gentileza e pelo atendimento atencioso de sempre.

A CAPES, pelo apoio financeiro, sem o qual esta tarefa seria muito mais difícil.

Ao dever cumprido, pois o dever é o mais belo prêmio da razão; dependente dele como o filho depende de sua mãe, o homem deve amar o dever, não porque o preserve dos males da vida, aos quais a Humanidade não pode se subtrair, mas porque dá à alma o vigor necessário ao seu desenvolvimento.

Obrigado DEUS, pela oportunidade de cumprir mais este dever!

"Sendo o pensamento criador a coisa mais importante que estabelece a diferença entre o homem e o macaco, deveria ser tratado como um bem mais precioso que o ouro e preservado com grande cuidado."

A. D. Hall,
A Methodology for System Engineering

RESUMO

Uma das primeiras fases da análise estrutural por elementos finitos é a criação do modelo geométrico. Este modelo deve representar fielmente a estrutura no que diz respeito a ângulos, dimensões e forma. Quando os objetos de estudo são estruturas biomecânicas, a dificuldade na realização das análises aumenta, primeiro porque os modelos considerados apresentam geometria quase sempre irregular, segundo pela dificuldade na realização de experimentos desta natureza, já que envolvem, em geral, organismos e tecidos vivos. Inserida, portanto, no contexto de Modelagem Científica Computacional, esta pesquisa procurou aplicar a computação a outras áreas do conhecimento, com o objetivo de criar modelos computacionais para situações em que é inviável testar ou medir as diversas soluções possíveis para um fenômeno a partir de modelos experimentais, neste caso, regiões ósseas. Para atingir este objetivo um programa computacional foi desenvolvido: o *bioMeshCreate*. O bioMeshCreate é uma aplicação final, independente, multi-plataforma, orientado a objeto, implementado em C++ com o auxílio de um toolkit de visualização científica, o VTK (*The Visualization Toolkit*), e seu objetivo principal é reconstruir estruturas ósseas tridimensionais a partir de seções planas, normalmente imagens médicas digitais obtidas por exames de tomografia computadorizada ou ressonância magnética. A aplicação também permite visualizar as fatias planas, gerar a malha sobre a superfície reconstruída, aplicar filtros de redução de elementos e suavização de superfícies e integrar o volume obtido com softwares de análise por elementos finitos, a fim de completar a análise sobre a estrutura. Ainda é possível exportar dados em formatos compatíveis com processos de prototipagem rápida e gerar modelos de realidade virtual, os quais podem ser distribuídos e visualizados em um browser. Testes com dados reais de pacientes foram realizados, e também comparações com outros sistemas, a fim de apontar as limitações e vantagens de seu emprego no processo de obtenção de modelos com o uso de técnicas de reconstrução 3D e geração de malha, confirmando a viabilidade de seu uso.

Palavras-chave: biomecânica, método dos elementos finitos, imagem (medicina), reconstrução tridimensional, geração de malha.

ABSTRACT

One of the first phases of the structural analysis by finite elements is to create a geometric model. This model must faithfully represent the structure in what is related to angles, dimensions and form. When the study objects are biomechanical structures, the level of difficulty in the accomplishment of the analyses increases, first because the considered models present geometry almost always irregular, second for the difficulty for the accomplishment of experiments of this nature, as they involve, in general, organisms and live tissues. Inserted, therefore, in the context of Scientific Computational Modeling, this research tried to apply the computation to other areas of knowledge, with the objective of creating computational models for situations in which testing or measuring the several possible solutions for a phenomenon starting from experimental models isn't viable, in this case, bone areas. To reach this purpose, a computational program was developed: the *bioMeshCreate*. The bioMeshCreate is a final application, independent, multi-platform, object-oriented, developed in C++ with the support of a toolkit for scientific visualization, the VTK (*The Visualization Toolkit*), and its main objective is to rebuild three-dimensional bone structures starting from plane sections, usually digital medical images obtained by exams of computed tomography or magnetic resonance. The application allows the visualization plane slices, to generate the mesh on the rebuilt surface, to apply filters of elements reduction and mesh smoothing and to integrate the volume obtained with finite elements analysis softwares, in order to complete the analysis on the structure. It is still possible to export data in compatible formats with rapid prototyping processes, and to generate virtual reality models, which can be distributed and visualized in a browser. Tests with patients' real data were accomplished, and also comparisons with other systems, in order to point the limitations and advantages of using this system in the process of obtaining models with the use of techniques of 3D reconstruction and mesh generation, confirming the viability of its use.

Keywords: biomechanics, finite element method, image (medicine), three-dimensional reconstruction, mesh generation.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

ANSI	American National Standards Institute
APDL	ANSYS Parametric Design Languade
API	Application Programming Interface
BMP	Bitmap
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CDT	Constrained Delaunay Triangulation
CG	Computação Gráfica
CT	Computed Tomography
DICOM	Digital Imaging Communications in Medicine
DOM	Document Object Model
EDPs	Equações diferenciais parciais
GIF	Graphics Interchange Format
GUI	Graphic User Interface
IGES	Initial Graphics Exchange Specification
ITK	Insight Toolkit
JPEG	Joint Photographic Experts Group
LER	Lesão por Esfoços Repetitivos
MC	Marching Cubes
MEF	Método dos Elementos Finitos
MITK	Medical Imaging Interaction Toolkit
MR	Magnetic Resonance
MRI	Magnetic Resonance Imaging
MT	Marching Tetrahedra
NIST	National Institute of Standards and Technology
NURBS	Nonuniform Rational B-Splines
OOP	Object Oriented Programming
PC	Personal Computer
PNG	Portable Network Graphics
PVL	Parâmetros do Volume
ROI	Region of Interest
RP	Rapid Prototyping
SPECT	Single Photon Emission Computed Tomography
STL	STereoLitography
VC	Visualização Científica
VRML	Virtual Reality Modeling Language
VTK	Visualization ToolKit
VV	Visualização Volumétrica
XML	eXtensible Markup Language
W3C	World Wide Web Consortium

1 INTRODUÇÃO

Uma das áreas de grande interesse hoje em dia tem sido a biomecânica, devido aos enormes benefícios que tem proporcionado ao homem. Dentro da biomecânica existem várias frentes de pesquisa e estudo, sendo este, voltado para a obtenção de modelos computacionais tridimensionais, mais precisamente modelos de regiões ósseas, para posterior análise por elementos finitos.

O forte interesse em entender e explorar o processo de reconstrução de imagens e geração de malhas está diretamente ligado à solução numérica de problemas científicos e de engenharia, principalmente para simulações de análise estrutural. Isto porque a eficiência de tais análises depende em grande parte da obtenção de um modelo geométrico que represente com fidelidade a estrutura analisada.

É com base nesta prerrogativa que todo este estudo está fundamentado e será apresentado ao longo dos próximos capítulos.

Justificativa

Com o intuito de solucionar problemas desta natureza, ou seja, a reconstrução tridimensional de regiões ósseas e geração de malhas sobre a estrutura obtida com a ajuda de um programa computacional, foi realizado um prévio estudo a fim de verificar as diferentes formas de atingir este resultado.

Desse modo, pelo que foi estudado, a solução poderia ser atingida por três técnicas distintas quanto ao sistema computacional envolvido. Em cada uma delas, o domínio da tecnologia adquirido ao longo do processo, o tempo de desenvolvimento e o grau de dificuldade apresentam níveis diferenciados. As vantagens e desvantagens de cada técnica serão elucidadas abaixo, podendo ser observadas de forma resumida na Figura 1.1.

A primeira técnica considerada foi a utilização de um software de reconstrução tridimensional e geração de malhas já existente para este fim. Esta solução tem baixa complexidade, pois uma vez adquirido o software, é necessário apenas entender seu funcionamento, partindo de imediato para a etapa de simulações e respostas. Oferece como vantagem o reduzido tempo de desenvolvimento, o que permite maior dedicação à análise e

comparação dos resultados, contudo não proporciona a aquisição de novos conhecimentos com respeito aos métodos de reconstrução de imagem e geração de malha. E, no caso de um produto comercial, ainda deve ser levado em conta o custo de aquisição de licenças, que acaba encarecendo o processo.

Outra técnica considerada foi a construção de um software próprio, utilizando componentes de código-fonte aberto, com alguns recursos numérico-computacionais já implementados. Esta abordagem garante maior aprendizado em relação à anterior, tendo em vista a possibilidade de realizar testes com os algoritmos disponíveis e escolher aquele que melhor se adapte à natureza dos problemas propostos. Porém, o tempo de desenvolvimento aumenta consideravelmente devido à etapa de construção do software, que envolve desde a escolha da tecnologia adequada, análise e planejamento, manipulação de estruturas de dados complexas (dados de imagem), tratamento de diferentes formatos de arquivo, até questões de eficiência como gerenciamento de memória, entrada/saída, e eficiência do código. Como conseqüência, a complexidade do processo aumenta proporcionalmente.

A última técnica analisada se refere à construção de um novo software e seus componentes a partir da implementação direta dos métodos numérico-computacionais de reconstrução tridimensional e geração de malha. Esta abordagem é a que oferece maior domínio sobre os métodos empregados, pois a implementação direta, a partir do algoritmo, exige pleno conhecimento dos métodos. Mas, devido à sua complexidade, esta abordagem requer muito mais tempo para ser desenvolvida. Além disso, a implementação pura e simples dos algoritmos existentes não representa efetiva contribuição, a menos que melhorias sejam acrescentadas.

		VANTAGENS / DESVANTAGENS		
		Domínio da Tecnologia Adquirido	Tempo de Desenvolvimento da Solução	Complexidade do Processo
METODOLOGIAS	**Uso de software de terceiro**	BAIXO	BAIXO	BAIXO
	Implementação de um novo software com a ajuda de componentes especializados	MÉDIO-ALTO	MÉDIO-ALTO	MÉDIO-ALTO
	Implementação de um novo software a partir do método numérico-computacional	ALTO	ALTO	ALTO

Figura 1.1 – Vantagens e desvantagens de cada técnica empregada no processo de reconstrução tridimensional, quanto ao software. / Fonte: O autor.

Com base na avaliação apresentada, a opção de desenvolver o próprio software para reconstrução tridimensional e geração de malhas, com o uso de componentes especializados, mostrou-se interessante. Pois, além de promover um bom aprendizado sobre o assunto, tem como resultado prático a obtenção de um produto final, com a perspectiva de ser aperfeiçoado em trabalhos subseqüentes a este. Visando desse modo, atender várias frentes de estudo envolvendo a elaboração de modelos computacionais para análise por elementos finitos, tanto na própria universidade como em parceria com outros centros de pesquisa.

Objetivos

Este trabalho teve como objetivo principal, a elaboração de um programa computacional capaz de reconstruir modelos ósseos tridimensionais a partir de seções planas, gerar a malha sobre a estrutura obtida e integrar os modelos com um software de análise por elementos finitos.

Para atingir este resultado, vários passos intermediários foram necessários. Entre eles, a exploração dos principais métodos de reconstrução de imagens tridimensionais a partir de fatias planas, e a investigação de métodos de geração de malha a partir de modelos tridimensionais.

Seguido por este levantamento prévio, foram pesquisados componentes especializados em tratamento de imagens, reconstrução 3D e geração de malha, com a finalidade de auxiliar no processo de desenvolvimento do programa computacional proposto.

Na fase de implementação, foi desenvolvido o software para reconstrução de modelos 3D e geração de malha, com os seguintes objetivos específicos:

- ler, interpretar e converter imagens obtidas a partir de tomografias computadorizadas ou ressonância magnética em formato médico digital;
- reconstruir volumes a partir das imagens lidas;
- gerar malha sobre a superfície reconstruída;
- aplicar filtros de redução de elementos e suavização de superfícies sobre a malha gerada;
- possibilitar a exportação para formatos compatíveis com softwares de análise por elementos finitos.

Por fim, na fase de discussões e resultados foram reconstruídas estruturas ósseas reais a partir de imagens tomográficas, procurando estabelecer comparações com o resultado de outras pesquisas e com o de outros sistemas.

Delimitação do Problema

Esta pesquisa teve como origem, um projeto envolvendo a análise estrutural sobre próteses e implantes odontológicos. Tal estudo foi realizado via modelagem e uso do método dos elementos finitos. Contudo, problemas referentes ao processo de construção do modelo dificultaram muito as análises, motivando assim a elaboração de um programa computacional que pudesse auxiliar na tarefa.

O programa foi então elaborado, tendo sempre em vista os objetivos específicos apontados. Logo, alguns recursos, embora pertinentes ao processo de reconstrução tridimensional e geração de malha, não foram desenvolvidos nesta versão, como por exemplo, a implementação de filtros para garantir a qualidade da malha gerada.

Embora as malhas de volume sejam de grande importância para o cálculo de esforços e tensões sobre estruturas biomecânicas pelo método dos elementos finitos, a implementação restringiu-se a um algoritmo de geração conhecido. Restrições de contorno ou refinamentos da malha não foram implementados, uma vez que o principal objetivo aqui é reconstruir imagens tridimensionais e gerar a malha de superfície. A geração da malha de volume constitui, portanto, um estudo inicial, sem a pretensão de abranger todo o assunto.

Também não fez parte desta primeira versão, o processo de isolamento de regiões de interesse e eliminação de ruídos da imagem. Nos casos em que isto se fez necessário, ferramentas auxiliares foram empregadas para completar o processo.

Contribuição

A principal contribuição dada por este trabalho foi, sem dúvida, o sistema computacional desenvolvido para reconstrução de imagens tridimensionais e geração de malhas. O software construído e denominado bioMeshCreate foi programado em C++ usando

o paradigma da orientação a objeto, em um ambiente de desenvolvimento livre. A aplicação foi desenvolvida com ferramentas multi-plataforma (Unix, Linux, Windows e Mac OS), garantindo assim maior portabilidade. No entanto, a primeira versão foi compilada e disponibilizada para a plataforma Windows 32-bits.

O bioMeshCreate reconstrói regiões ósseas a partir de fatias planas no formato médico digital, gera a malha de superfície e permite aplicar filtros para reduzir o número de elementos da malha e melhorar sua aparência. Facilita a integração dos modelos gerados com softwares de análise por elementos finitos. Além disso, permite exportar os modelos para vários formatos, incluindo formatos de imagem, dados poligonais, malha de triângulos e modelos de realidade virtual.

Constitui, portanto, a base para trabalhos na área de construção de modelos computacionais para análise por elementos finitos.

Organização

O *Capítulo 2 – Revisão da Literatura* apresenta conceitos relativos à modelagem e visualização científica, elucida alguns métodos de reconstrução de imagem e geração de malha, em seguida descreve, em linhas gerais, os principais toolkits e ferramentas de desenvolvimento voltadas para este fim.

O desenvolvimento inicia-se com o *Capítulo 3 – Técnicas de Reconstrução e Geração de Malha*, onde são tratados os métodos numérico-computacionais utilizados na fase de implementação. O *Capítulo 4 – O VTK*, aborda este toolkit de desenvolvimento orientado a objetos que foi adotado para a implementação computacional nesta pesquisa. O *Capítulo 5 – O Programa Computacional bioMeshCreate*, apresenta esta aplicação, produto final do trabalho, onde são descritos sua arquitetura, organização, e os recursos relacionados à reconstrução tridimensional, geração de malha e elementos finitos.

Os resultados são mostrados a partir do *Capítulo 6 – Aplicações a Estruturas Biomecânicas*, onde é feito um caso de uso completo do programa bioMeshCreate sobre uma região óssea, desde a sua reconstrução a partir de imagens planas, até a geração, simplificação e suavização da malha de superfície, sua exportação para arquivo e posterior importação por um software de análise por elementos finitos. Já o *Capítulo 7 – Análise Comparativa dos Resultados* estabelece uma comparação com outras pesquisas, apontando vantagens e

desvantagens, e em seguida comparando, em termos quantitativos, o desempenho do bioMeshCreate com o de outros sistemas, já validados e testados.

A conclusão, através do *Capítulo 8 – Conclusões e Trabalhos Futuros* faz uma síntese daquilo que foi realizado, aponta as limitações e vantagens das técnicas empregadas e sugere novas implementações, visando o aperfeiçoamento e melhoria do programa computacional bioMeshCreate, dentro daquilo a que se propõe a fazer.

2 REVISÃO DA LITERATURA

Com base na revisão da literatura realizada, o presente capítulo aborda conceitos referentes à modelagem computacional e visualização científica, alguns métodos de reconstrução de imagem e geração de malha conhecidos. E para finalizar, apresenta algumas bibliotecas e toolkits de desenvolvimento relacionados ao assunto.

2.1 MODELAGEM E VISUALIZAÇÃO CIENTÍFICA

> *"Obtenção de modelos computacionais tridimensionais (regiões ósseas) para análise por elementos finitos"*

A frase acima encerra a idéia central deste estudo. De fato o que se está procurando é uma solução computacional aproximada para o problema.

A solução computacional aproximada, é baseada em representações simplificadas que descrevem idealmente a estrutura física e o comportamento mecânico de tais objetos. A estas representações dá-se o nome de *modelos*.

Na literatura existem vários conceitos para modelos. Por exemplo, Mäntyla (1988), define um modelo como sendo:

> Um objeto construído artificialmente que torna a observação de outro objeto mais fácil. Para tornar a observação possível, modelos físicos de objetos tridimensionais tais como edifícios, naves e carros, usualmente compartilham as dimensões relativas e a aparência geral do objeto físico correspondente, mas não o tamanho.
>
> [...] Modelos matemáticos, amplamente utilizados em vários campos da ciência e engenharia, representam alguns dos aspectos comportamentais de fenômenos modelados em termos de dados numéricos e equações.

A busca por modelos computacionais que possam representar objetos do mundo físico criou uma onda de interesse muito grande nos últimos anos, onde várias pesquisas têm sido realizadas. Não obstante, a idéia de representar modelos através de imagens data de muito tempo atrás. Por exemplo, Brodlie (1995) em seu trabalho "*Visualização científica – passado, presente e futuro*" mostra alguns casos de uso de técnicas de visualização usadas antes do computador ser inventado. Ele dá o exemplo da terra, onde a inclinação da órbita planetária com respeito ao tempo é desenhada como um conjunto de gráficos em série de

tempo. E acredita-se ter sido feito há dez séculos, certamente em um período pré-Galileu, uma vez que o 'Sol' era tratado como um dos planetas.

Hoje, com o advento da computação, os modelos procurados podem ser traduzidos em imagens e adequadamente representados em um computador. Este trabalho preocupa-se, portanto, com a geração, manipulação e análise dessas imagens, através do computador. Este tipo de atividade caracteriza uma área da Ciência da Computação conhecida como *Computação Gráfica* (CG) (MANSSOUR;COHEN, 2006). Atualmente a computação gráfica pode ser considerada como uma área de fronteira, envolvendo várias disciplinas científicas.

A percepção do relacionamento *modelo* ↔ *imagem* é um ponto chave na contextualização deste estudo. A Figura 2.1 abaixo ilustra esse relacionamento:

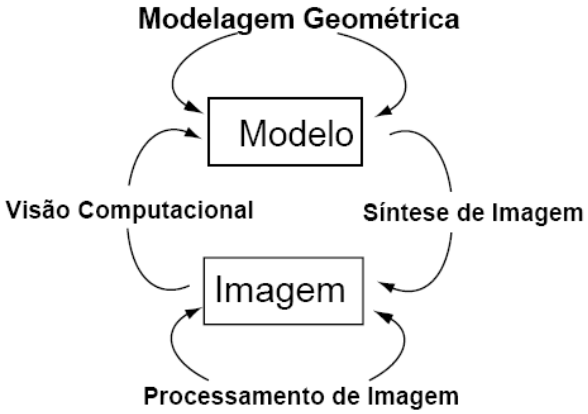


Figura 2.1 – Relacionamento modelo ↔ imagem.
Fonte: Velho e Gomes (2001).

Segundo Velho e Gomes (2001), a *Síntese de Imagem* faz a transformação de modelos geométricos em imagens digitais. A transformação inversa denomina-se *Análise de Imagens*. Os processos que atuam exclusivamente sobre um dos tipos de dados (modelo ou imagem) para criá-los ou para modificá-los correspondem respectivamente à *Modelagem Geométrica* e ao *Processamento de Imagens*. O processamento de imagens é necessário em quase todo o tipo de manipulação com imagens e foi utilizado na parte prática dessa pesquisa durante a etapa de implementação.

Mais especificamente este estudo está voltado para a *Computação Gráfica 3D*, que trata o problema da modelagem e síntese de imagens em cenas tridimensionais. Esta é uma das áreas mais complexas e importantes da CG, que neste caso está direcionada para a *Visualização Científica* (VC).

A visualização científica é utilizada para a visualização de simulações e estruturas complexas em diversas disciplinas científicas, tais como Matemática, Medicina, e Biologia.

O presente trabalho utiliza, portanto, recursos de VC para reconstruir imagens tridimensionais (regiões ósseas) a partir de fatias planas (imagens tomográficas), representando-a através de uma malha poligonal (modelo computacional). Em seguida, o modelo é exportado para um formato que permite a integração com um software de análise por elementos finitos, a fim de ser posteriormente processado (análise pelo MEF).

Para obter o modelo computacional foi implementado um software. As etapas que aparecem por trás dessa implementação, segundo Gomes e Velho (1998), podem ser melhor entendidas através de um paradigma de abstração que consiste em estabelecer quatro universos (conjuntos): o universo físico **F**, o universo matemático **M**, o universo de representação **R**, e o universo de implementação **I**, conforme pode ser observado na Figura 2.2. Este paradigma é denominado *paradigma dos quatro universos*.

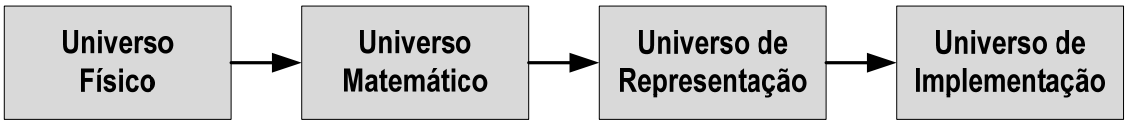


Figura 2.2 – O paradigma dos quatro universos.
Fonte: Gomes e Velho (1998).

Segundo os autores, o *universo físico* contém os objetos do mundo real que se pretende estudar. O *universo matemático* contém uma descrição abstrata dos objetos do mundo físico. O *universo de representação* é constituído por descrições simbólicas e finitas associadas a objetos do universo matemático. No *universo de implementação* são associadas as descrições do universo de representação às estruturas de dados, com a finalidade de obter uma representação do objeto no computador. O objetivo do universo de implementação é separar a etapa de discretização (representação), das particularidades da linguagem de programação utilizada.

Nesta pesquisa o universo físico pode ser entendido como a região óssea a ser reconstruída. O universo matemático dependerá do método empregado na reconstrução tridimensional. No caso do *Marching Cubes*, por exemplo, dado um valor limiar **c**, o algoritmo produz uma aproximação triangular para a superfície, através da função $S:\Re^3 \to \Re$, tal que $S_F(c) = \{(x,y,z) : F(x,y,z) = c\}$. Esta função escalar está associada ao universo matemático. O universo de representação é dado pelo domínio dos pontos posicionados sobre uma grade retilínea tridimensional. E finalmente, o universo de implementação, que neste caso é constituído por um *dataset* (conjunto de dados) capaz de armazenar a representação do objeto no computador.

2.2 MÉTODOS DE RECONSTRUÇÃO TRIDIMENSIONAL

Os métodos de reconstrução tridimensional estão inseridos em uma subárea da visualização científica denominada *Visualização Volumétrica* (VV). A visualização volumétrica (ou de volumes) é o termo usado para todos os possíveis modos de se representar um conjunto de dados tridimensional em uma região bidimensional, tipicamente, a tela de um computador. Isto está relacionado à representação, manipulação e renderização de dados volumétricos (LICHTENBELT, 1995).

Os algoritmos de visualização podem ser subdivididos em duas categorias principais: técnicas de reconstrução de volumes e técnicas de reconstrução de superfícies.

Reconstrução de Volumes. Esta categoria de reconstrução varre todo o volume de dados, buscando gerar uma imagem do objeto a partir de alguns parâmetros pré-definidos, tais como densidade, opacidade, textura, campos de vetores normais, etc. Tais parâmetros indicam como os dados internos do volume são combinados e processados para gerar a imagem final, através da interação com a luz. Estes métodos de reconstrução são usados principalmente para visualizar objetos amorfos como gases, dados geológicos e até mesmo estruturas humanas sem forma definida como sangue e gordura. Porém nada impede que estes métodos também sejam utilizados para visualizar objetos com uma morfologia bem definida. Um dos problemas desta categoria de reconstrução é a grande quantidade de dados processados (PEIXOTO; GATTASS, 2000).

Reconstrução de Superfícies. Também chamados de extração de isosuperfícies, esta categoria utiliza técnicas de extração de contorno para separar similaridade de cores em regiões distintas. Quando os dados são contornados, está se construindo efetivamente o contorno entre estas regiões. Esses contornos correspondem a curvas no plano[1]. A partir da definição desses contornos é aplicado algum método de interpolação para reconstruir a superfície do objeto. Esta categoria é utilizada principalmente para reconstruir superfícies de objetos morfologicamente bem definidos, como por exemplo, imagens médicas correspondentes a partes do corpo, como pele, ossos e outros órgãos. Uma vantagem desta categoria é que não requer tantos recursos de armazenamento, uma vez que somente os contornos da estrutura são processados para reconstruir o volume.

Entre os métodos de reconstrução de superfícies mais conhecidos encontram-se o

[1] Contornos tridimensionais são chamados de isosuperfícies e podem ser aproximados por diferentes tipos de primitivas poligonais.

Contour Connecting (KEPPEL, 1975), o *Marching Cubes* (LORENSEN; CLINE, 1987), o *Dividing Cubes* (CLINE et al., 1988) e o *Marching Tetrahedra* (SHIRLEY; TUCHMAN, 1990). Entre os de reconstrução de volumes figuram o *Ray-Casting* (TUY; TUY, 1984), o *Ray-Merging* (LEVOY, 1990), o *Shear-Warping* (LACROUTE; LEVOY, 1994), entre outros.

A eficácia relativa aos métodos de reconstrução: de superfícies ou de volumes tem sido alvo de investigações já há algum tempo (UDUPA; ODHNER, 1993), mas baseado no que foi exposto e por atender de maneira satisfatória os objetivos propostos, este estudo restringiu-se à utilização de métodos de reconstrução de superfícies. A seguir será apresentado um panorama dos trabalhos relacionados à evolução dessa categoria de métodos.

2.2.1 Métodos de Reconstrução de Superfícies

O primeiro trabalho encontrado foi o de Keppel (1975), denominado *Contour Connecting* que descreve um algoritmo para encontrar uma aproximação ótima, usando triângulos, de uma superfície definida por um conjunto de linhas de contorno. Este método aplica-se sem restrição a quaisquer linhas de contorno, dentro de regiões convexas ou côncavas. Ele consiste em assumir uma simples função adequada para a manipulação dos problemas de otimização. Este esquema combinatorial da triangulação então permite a resolução do problema com técnicas conhecidas de teoria dos grafos.

Após o método de conexão de contornos de Keppel, merece destaque o algoritmo proposto por Fuchs, Kedem e Uselton (1977) que segue a mesma linha de raciocínio de seu antecessor, porém ao contrário de Keppel que introduziu uma redução do problema encontrando uma trajetória dirigida por grafos e usou certa heurística para escolher a trajetória apropriada, este novo método também reduziu o problema em teoria dos grafos, mas não utilizou qualquer heurística, permitindo assim, um desenvolvimento geral com uma variedade de opções possíveis para escolha de critérios de otimização.

Ainda baseado em decisões heurísticas e teoria dos grafos, Ganapathy e Dennehy (1982) apresentaram um novo método com o objetivo de ser usado no campo da avaliação de ultra-som não-destrutivo, para produzir imagens mostrando imperfeições em vasos de pressão. A heurística utilizada, segundo eles, era totalmente generalizada, robusta e usual em outras aplicações como sistemas de visão industrial. Desse modo, Ganapathy apresentou uma nova heurística para triangulação de superfícies 3D formadas pela abrangência de um

conjunto de contornos planos. Esses contornos podiam ter geometrias arbitrárias, mas eram restritos ao plano.

Em 1987 era publicado um artigo por Lorensen e Cline (1987) onde o clássico algoritmo de *Marching Cubes* surgia. Este algoritmo foi sofrendo vários ajustes e melhorias ao longo das duas últimas décadas, sendo até hoje usado em aplicações relacionadas à extração de isosuperfícies de modelos sólidos com contornos bem definidos, por ser de fácil entendimento e implementação, alcançando resultados satisfatórios, com baixo custo operacional e de armazenamento. Por estes motivos o algoritmo proposto por Lorensen e Cline acabou sendo utilizado nesta pesquisa durante a fase de implementação computacional. O capítulo seguinte detalha o funcionamento deste algoritmo, por hora será feito apenas uma apresentação do mesmo e discutido alguns trabalhos que modificaram o algoritmo original visando obter melhores resultados.

O Marching Cubes (MC) foi criado como parte das pesquisas da General Eletric Company a fim de processar imagens médicas em 3D provenientes de tomografia computadorizada (CT), ressonância magnética (MR) e tomografia computadorizada com emissão simples de fótons (SPECT). Esse algoritmo, na época em que fora criado, não tinha precedentes nesta mesma linha, estando essencialmente focado no uso de uma tabela de casos de interseção de arestas para descrever como uma superfície corta um cubo em um conjunto de dados 3D. A configuração inicial apresentava 15 casos topologicamente diferentes. Além disso, realismo adicional era atingido pelo cálculo, a partir dos dados originais, do gradiente normalizado. O sombreamento pelo gradiente contribuía para a qualidade da imagem dando contraste que dependia da orientação da superfície. A estrutura poligonal resultante podia ser exibida em qualquer sistema gráfico convencional da época.

Os mesmos autores, Cline et al. (1988) formalizaram outra técnica de reconstrução denominada *Dividing Cubes*, onde pela subdivisão do voxel, eles interpolavam os dados em três dimensões para aproximar a função de densidade contínua com uma fina treliça cúbica de inteiros. Esta subdivisão incrementou a precisão da interpolação. Todavia, a fina subdivisão, implicava em mais pontos na superfície a serem processados.

Dois trabalhos diretamente relacionados ao método-MC mereceram destaque. O primeiro escrito por Chernyaev (1995) ficou bastante conhecido, pois este pesquisador verificou a existência de falhas no método, ocasionado pela possibilidade de pequenos "buracos" que apareciam como resultado da discrepância na conexão dos vértices nas faces compartilhadas de duas células adjacentes. Outro problema que Chernyaev detectou foi com relação aos casos de ambigüidade, onde isosuperfícies com topologias diferentes atendendo às

solicitações do método poderiam produzir resultados diferentes. Foi então que, após um estudo mais detalhado desses casos, ele propôs aumentar o número inicial de topologias distintas, de 15 para 33, a fim de resolver tais problemas, ficando o seu método conhecido como MC-33.

O segundo trabalho foi escrito por Hege et al. (1997), o qual publicou uma modificação no algoritmo-MC, denominado: "*Um algoritmo generalizado de Marching Cubes baseado em classificações não-binárias*". O tradicional método-MC usava classificação binária para identificar os casos, já nessa abordagem um número arbitrário de classes de vértices podem ser especificados. Similar ao algoritmo-MC todas as células da grade são atravessadas e classificadas de acordo com o número de diferentes classes de vértices envolvidos e suas combinações. A solução para cada configuração é calculada com base em um modelo que atribui probabilidades aos vértices e os interpola. Um método automático para encontrar a triangulação é introduzido, o qual aproxima as superfícies de contorno – implicitamente definidas pelo modelo – de um modo topologicamente adequado. Tabelas de casos garantem o bom desempenho do algoritmo. O propósito deste algoritmo era fornecer solução robusta e única, evitando as ambigüidades que ocorriam em outros métodos.

Vários outros trabalhos foram publicados visando a reconstrução 3D por extração de isosuperfícies. Por exemplo, o trabalho de Shirley e Tuchman (1990) usava tetraedros para resolver problemas de ambigüidade e compor as superfícies, porém com maior custo computacional, uma vez que o número de combinações aumentava consideravelmente neste caso.

Na verdade o *Marching Tetrahedra* (MT) também é uma variação do MC, segundo Treece, Praguer e Gee (1998), que em seu trabalho "*Marching tetrahedra regularisado: extração de isosuperfícies*" modificou o algoritmo-MT a fim de fornecer melhor aspecto de proporcionalidade aos triângulos da malha gerada, através de técnicas de simplificação, podendo chegar a uma malha reduzida a 70% em relação ao método-MT original.

Ekoule, Peyrin e Odet (1991) propuseram outro algoritmo para resolver problemas que ocorrem quando os contornos são não-convexos ou quando os contornos em duas fatias sucessivas são muito diferentes. Do mesmo modo, a presença de múltiplos contornos em uma fatia deixa ambigüidades na definição das ligações apropriadas. Ekoule propôs uma solução para problemas desta natureza definindo um procedimento geral de triangulação baseado em uma simples heurística. Seu algoritmo aplica-se à reconstituição de superfícies complexas e como exemplo ele fez a reconstrução de uma vértebra da coluna humana.

Os pesquisadores Klemt e Infantosi (2000) apresentaram um método que consiste em

criar *Unidades Espaciais*, que dividem o espaço em regiões de igual volume e permite organizar as faces segundo características comuns, como posição espacial e normal à face. A partir da informação sobre o(s) plano(s) de dissecção (qualquer inclinação) e do lado a ser retirado, as faces são classificadas em *dissecadas*, *mantidas* e *atravessadas*, até que não existam mais faces *atravessadas*. Assim, obtêm-se um conjunto de arestas criadas pela dissecção que formam o contorno da superfície de dissecção. Os resultados, utilizando o crânio humano e mandíbula, mostraram que o método de visualização de superfícies proposto possibilita a exploração do interior, através da simulação de dissecção por planos, o que geralmente só é atribuído aos métodos de visualização de volumes.

Outro trabalho que trata da reconstrução 3D por isosuperfícies é o de Barequet, Shapiro e Tal (2000) onde vários casos de uso de sua técnica são ilustrados. De modo resumido, seu algoritmo consiste de dois passos principais. Primeiro, similares porções do contorno de cada par de fatias consecutivas são coincididos e preenchidos. Os polígonos ficam então em três dimensões e são chamados de fissuras (clefts). Em seguida, as fissuras são consistentemente trianguladas com as interpolações nas camadas vizinhas.

2.2.2 Aplicações das Técnicas de Reconstrução de Superfícies

Várias aplicações dos métodos de reconstrução foram encontrados na literatura e alguns serão comentados abaixo.

Kardos, Hajder e Chetverikov (2005) fizeram uma apresentação utilizando *Fast Marching Methods* (SETHIAN, 1998), com os mesmos interesses desta pesquisa, ou seja, a reconstrução de superfícies ósseas a partir de imagens de CT/MR.

Poh e Kitney (2005) implementaram uma ferramenta de segmentação e reconstrução 3D que disponibiliza os modelos obtidos ao usuário através da internet, lançando mão de conceitos relacionados à realidade virtual (VRML). Um tutorial que trata as potencialidades da VRML e suas aplicações no campo da visualização científica pode ser encontrado em Kageyama e Ohno (2005).

Outra aplicação da reconstrução de modelos tridimensionais, envolvendo a área médica merece destaque: a construção de protótipos biomédicos baseado em técnicas de RP (prototipagem rápida). Entre os trabalhos encontrados nesta área, figuram: Souza et al. (2001), Souza, Centeno e Pedrini (2003), Milovanović e Trajanović (2007) e Meurer et al. (2008).

Ao se abordar técnicas de RP surge o formato associado, STL (STereoLitography). Mais detalhes sobre este formato podem ser encontrados no trabalho de Kumar e Dutta (1997).

Síntese dos Métodos de Reconstrução de Superfícies e suas Aplicações

O diagrama abaixo (Figura 2.3) contempla todos os métodos de reconstrução de superfícies revisados, bem como as suas aplicações. Seu objetivo é dar um panorama geral dessa categoria de métodos. As conexões entre os blocos indicam a existência de características comuns entre os métodos.

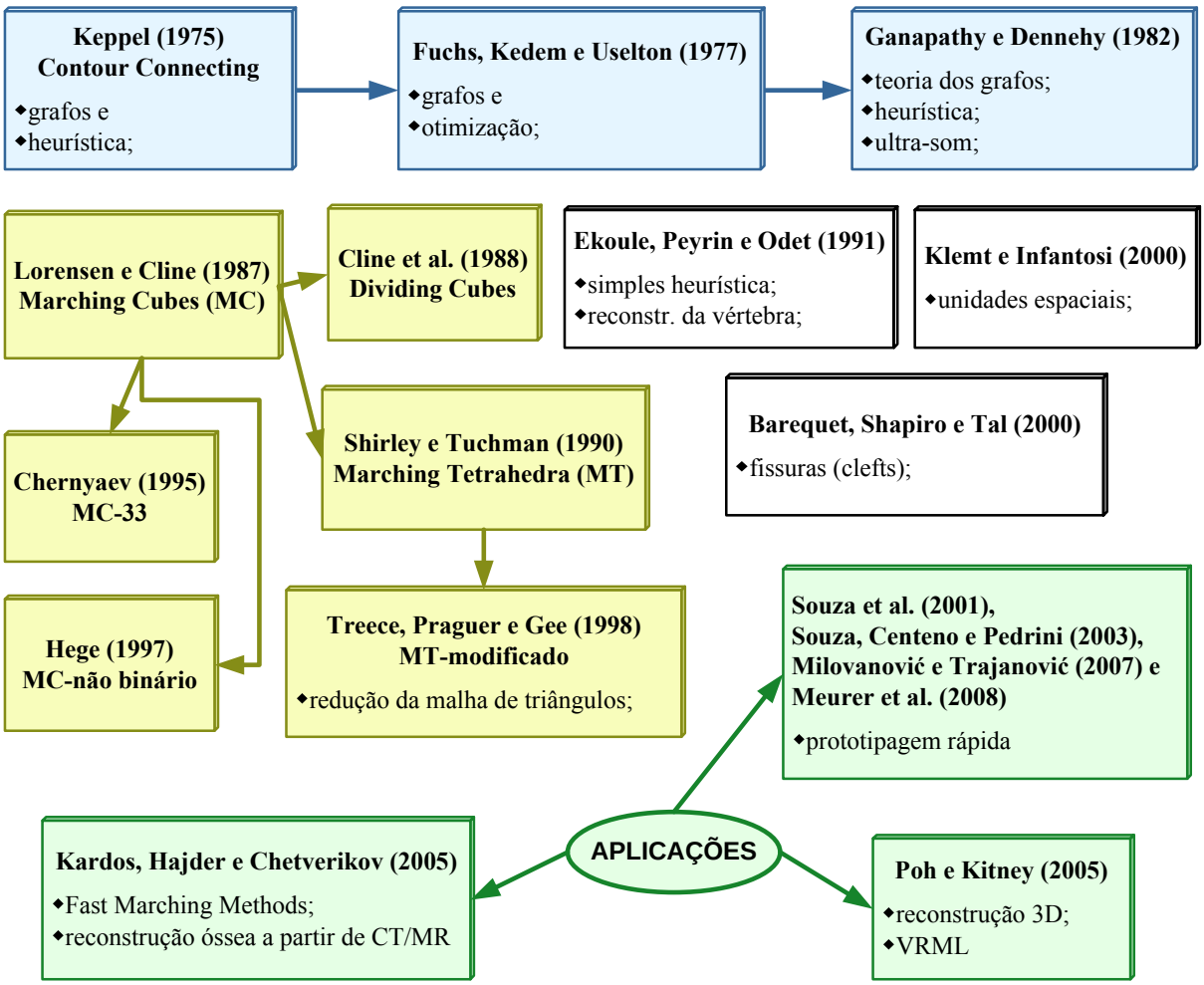


Figura 2.3 – Os métodos de reconstrução de superfícies revisados e algumas aplicações.
Fonte: O autor.

2.3 MÉTODOS DE GERAÇÃO DE MALHA

Quanto aos métodos de geração de malha, existem aqueles que geram elementos planos poligonais (normalmente triângulos), podendo ser gerados sobre conjuntos de dados bidimensionais (figuras planas) e/ou sobre conjuntos de dados tridimensionais (figuras espaciais), formando neste último, uma casca (shell) ou malha de superfície. Os métodos de reconstrução tridimensional apresentados anteriormente, por exemplo, que compõem o volume pela sobreposição seqüencial das fatias planas, ao ligar os vértices adequadamente, acabam formando uma malha de superfície. Outros exemplos de geração de malha de superfície podem ser vistos em Lau e Lo (1996) e em Kocharoen et al. (2005).

Existem ainda os métodos geradores de malha que tomam um conjunto de dados necessariamente tridimensional e produzem elementos também tridimensionais: os poliedros (normalmente tetraedros). Nesse tipo de malha, também chamada malha de volume, os elementos 3D gerados costumam preencher todo o interior do volume. A qualidade desse tipo de malha é medida justamente pelo total preenchimento, sem sobreposição de elementos e sem deixar buracos (holes) na região interna, bem como pela distribuição homogênea dos elementos, isto quer dizer que os elementos devem ser poliedros convexos o mais regular[2] possível, como por exemplo, tetraedros regulares, hexaedros regulares, e assim por diante, dependendo do tipo de malha, e terem dimensões mais próximas quanto possíveis uns dos outros.

A maioria das técnicas encontradas neste levantamento bibliográfico utiliza a triangulação de Delaunay 3D para a geração da malha de volume, ou uma combinação desta com outros algoritmos, de modo a garantir a qualidade da malha gerada. Cabe lembrar ainda que o algoritmo de Delaunay, com o tempo foi sofrendo restrições e refinamentos visando atender novas necessidades, tal como a geração de malha sobre superfícies côncavas.

A seguir serão citados alguns métodos que empregam a triangulação de Delaunay como parte do processo de geração de malha de volume, uma vez que os conceitos pertinentes à triangulação, os principais algoritmos, restrições e refinamentos serão discutidos em capítulo separado. Por último, serão apresentados alguns artigos onde outras técnicas são empregadas na geração de malha volumétrica.

---

[2] Um poliedro convexo é regular quando suas faces são polígonos regulares e congruentes, e seus ângulos poliédricos são congruentes.

2.3.1 Geração de Malha com Base na Triangulação de Delaunay

O livro de Geometria Computacional de Berg et al. (2008) trata vários conceitos relacionados à triangulação de Delaunay. Outro trabalho nesta mesma linha é o de Bern e Eppstein (1995), nele os autores discutem triangulação ótima de domínios geométricos em duas e três dimensões. Uma triangulação ótima é uma partição do domínio em triângulos ou tetraedros que estão de acordo com alguns critérios de medida de tamanho, área, ou número de triângulos. É dado também uma visão geral sobre algoritmos heurísticos usados em alguns geradores de malha práticos.

Fang e Piegl (1993) implementaram a triangulação de Delaunay no plano usando malha uniforme. Kolingerová e Žalik (2002) sugerem duas melhorias para o algoritmo também para conjuntos de dados em duas dimensões. A primeira aumenta a velocidade sem requerimento adicional de memória. A segunda diminui os requerimentos de memória com uma pequena perda de desempenho.

Entre os algoritmos de triangulação de Delaunay em três dimensões existentes, foram pesquisados os de Bowyer (1981), Watson (1981) e Fang; Piegl (1995) que apresentam os detalhes da implementação. Kanaganathan e Goldstein (1991) realizaram uma análise comparativa entre quatro algoritmos de Delaunay para geração de malha tridimensional, apontando o algoritmo de Watson como o melhor em termos de qualidade de malha gerada. Su e Drysdale (1995) classificaram os algoritmos para construção da triangulação de Delaunay em três grupos básicos: *divide-and-conquer*, *sweepline* e *incremental*, e analisaram as características de cada grupo.

Desde que o algoritmo de Delaunay fora criado, várias modificações surgiram para resolver problemas relacionados à concavidade e geometrias complexas. Estas modificações ficaram conhecidas como CDT (constrained Delaunay triangulation) ou triangulação de Delaunay restrita. Foram criados também os refinamentos, estes mais voltados para a otimização, velocidade e robustez do algoritmo.

Dentre os algoritmos que implementaram restrições e refinamentos em duas dimensões para resolver problemas relacionados à concavidade e geometria irregular, foram encontrados os de Chew (1993) e Ruppert (1995). Sendo este último, bastante citado na literatura e um dos mais usados, constituindo uma referência com todos os passos necessários para a implementação. O CDT desenvolvido por Anglada (1997) resolve, além de problemas de concavidade, possíveis buracos encontrados na superfície. Outro CDT avaliado foi o de

Chin e Wang (1998). Magalhães, Passaro e Abe (2000) desenvolveram uma aplicação orientada a objeto, a qual exemplifica a implementação de um CDT.

Com relação aos refinamentos em três dimensões, foram estudados os de George, Hecht e Saltel (1990) que faz um estudo envolvendo geração automática de malha 3D com contorno da malha prescrito. Schroeder, Geveci e Malaterre (2004) fizeram um refinamento para resolver casos de ambigüidade na triangulação, ocasionado por degenerações, utilizando uma técnica de ordenação dos simplexos. Outras duas técnicas bastante recentes são as de Alliez et al. (2005), que usa além da triangulação de Delaunay restrita a superfícies de contorno, conceitos de energia variacional para melhorar a entrada, distribuindo os vértices de forma equilibrada na região interna da imagem, e Kolingerová e Žalik (2006) que reconstrói domínios com contornos dados por um conjunto de pontos, usando uma técnica heurística.

Quanto aos que se dedicaram a avaliar e melhorar aspectos de qualidade da malha gerada temos Li (2000) que se dedicou a resolver problemas de proporcionalidade e slivers (fatias finas), e o trabalho de Du e Wang (2006) que faz um estudo relacionado a recentes progressos em robustez e qualidade de geração de malha de Delaunay.

A próxima seção se atém aos algoritmos de geração de malha volumétrica que utilizam outras técnicas para constituir a malha.

2.3.2 Geração de Malha com Base em Outras Técnicas

Uma técnica baseada em tetraedrização de volumes por intervalo foi descrita por Nielson e Sung (1997). Esta técnica é uma generalização de isosuperfícies comumente associada com o algoritmo de Marching Cubes. Ele assume que a função $F(x,y,z)$ representa o domínio dos pontos posicionados sobre uma grade retilínea. Dado um valor limiar c, o algoritmo MC produz uma aproximação triangular para a superfície $S_F(c) = \{(x,y,z) : F(x,y,z) = c\}$. O intervalo do volume é definido como $I_F(\alpha,\beta) = \{(x,y,z) : \alpha \le F(x,y,z) \le \beta\}$. A técnica proposta por Nielson é relativamente simples e pode ser descrita em dois passos: Passo 1) cada voxel é decomposto em tetraedros e assume-se que F varia linearmente sobre cada tetraedro; Passo 2) Para cada tetraedro de cada voxel, o volume do intervalo é computado e decomposto em novos tetraedros. Os detalhes subjacentes a cada passo estão detalhados no artigo. Importante ressaltar que o algoritmo de

Nielson não utiliza a triangulação de Delaunay em nenhuma das etapas de seu desenvolvimento.

Uma classificação dos algoritmos de geração de malhas tetraédricas foi dada por Chae e Lee (1999). Segundo eles, os algoritmos podem ser classificados em três grupos: os que usam técnicas de *advancing-front*; os baseados em *Delaunay*; e os baseados em *octree*. O artigo propõe também um algoritmo para geração de malhas tetraédricas a partir de uma série de seções planas. Neste esquema, as seções planas e a superfície lateral entre duas seções são trianguladas primeiro, e então um algoritmo de advancing front é empregado para construir elementos tetraédricos pelo uso do que eles denominaram operadores básicos (*trimming*, *wedging*, *digging*, *finishing* e *splitting*). Alguns exemplos de malha foram construídas a partir das seções planas e analisadas com o software Adina.

No trabalho de Zhang, Bajaj e Sohn (2003) um algoritmo foi proposto para a extração de malhas adaptativas diretamente de dados de imagens volumétricas. A malha extraída pode ser tetraédrica ou hexaédrica e extensivamente usada para simulações em análise por elementos finitos. A técnica de Zhang combina filtros de difusão bilateral e anisotrópica com técnicas baseadas em espectro de contorno, isosuperfícies e seleção de volumes por intervalo. Depois disso, uma subdivisão octree de cima para baixo conjugada com um método de contorno dual é usada para extrair rapidamente a malha adaptativa tridimensional de elementos finitos. Na parte final do trabalho, Zhang exibe alguns exemplos de malha gerada, em geral com boa qualidade, no entanto, a combinação dos vários algoritmos necessários para a obtenção da malha acabam tornando o processo de implementação complicado, embora, segundo os autores, um programa extrator tenha sido implementado e executado em um PC comum.

A última técnica de geração de malha a ser apresentada é um algoritmo para geração de malha tridimensional baseado em imagens não-uniformes para aplicações médicas. Este gerador foi desenvolvido por Berti (2004) e tem por base o uso de *spacetrees* (octrees generalizados para dimensões arbitrárias) para construir uma representação hierárquica das informações da imagem. O algoritmo funciona em dois passos: primeiro, uma representação hierárquica da geometria (imagem) é construída, resultando em um spacetree, o qual determina a resolução da malha. Em um segundo passo, esse spacetree pode ser construído dentro de uma malha adaptada, resultando ou em uma triangulação ou em uma malha híbrida dominante.

Síntese dos Métodos de Geração de Malha Estudados

Os próximos diagramas encerram os métodos de geração de malha discutidos neste trabalho. Na Figura 2.4 é mostrada a classificação dos algoritmos de geração de malha segundo Chae e Lee (1999), são citados os algoritmos pesquisados que não usam a triangulação de Delaunay (advancing-front e octree), e também a subdivisão dos algoritmos de Delaunay segundo Su e Drysdale (1995).

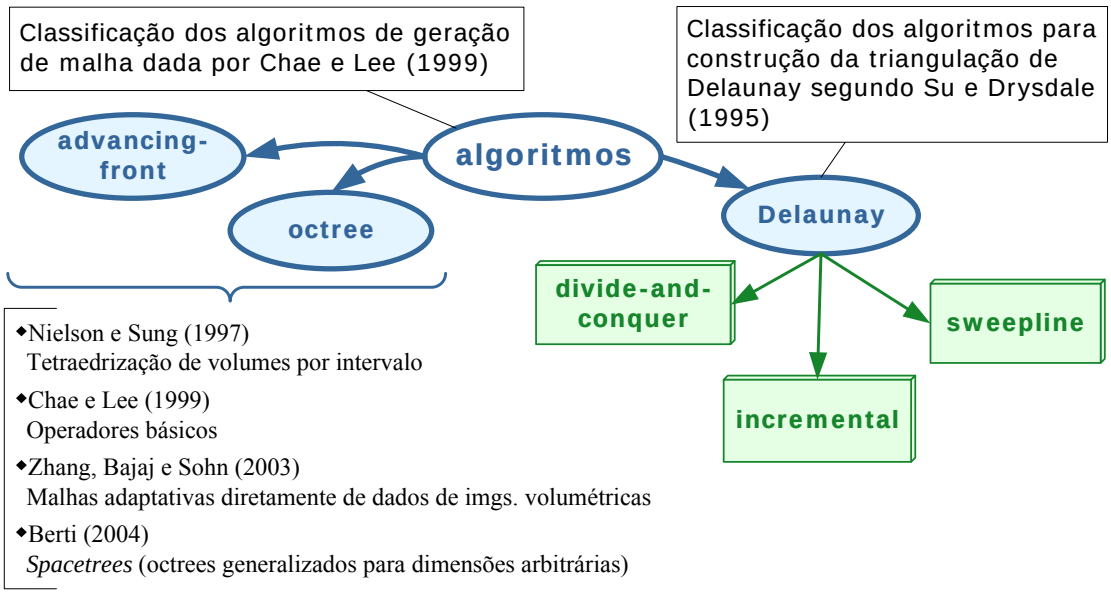


Figura 2.4 – Classificação dos métodos de geração de malha revisados.
Fonte: O autor.

Já na Figura 2.5 são apresentados os algoritmos de geração de malha baseados na triangulação de Delaunay. Os que aparecem em destaque são aqueles que apresentaram melhores resultados e serão tratados mais adiante, no próximo capítulo.

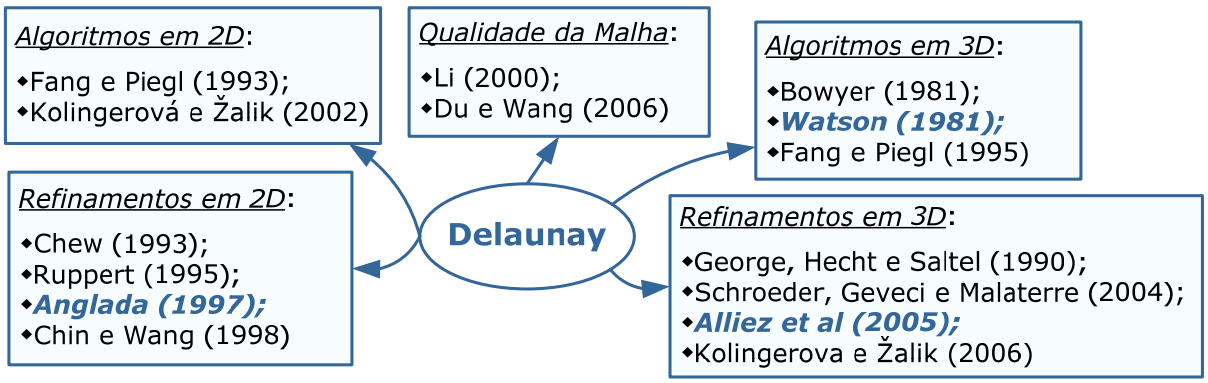


Figura 2.5 – Os métodos de geração de malha baseados na triangulação de Delaunay revisados.
Fonte: O autor.

2.4 TOOLKITS DE VISUALIZAÇÃO CIENTÍFICA

Durante o levantamento bibliográfico foram pesquisadas ferramentas com o objetivo de facilitar a tarefa de desenvolvimento do software proposto. Algumas bibliotecas gráficas e toolkits (pacotes) de desenvolvimento encontrados foram testados, e serão discutidos a seguir.

Um software conhecido em Engenharia é o MATLAB (2007). Este software é multi-plataforma e possui um conjunto de bibliotecas destinadas ao tratamento de imagens, denominado *Image Processing Toolbox*. Os recursos dessa toolbox são amplos, podendo-se destacar a análise de imagens, incluindo detecção, segmentação e medição, transformações espaciais, registro e suporte para processamento de imagens multidimensionais. Além disso, possui ferramentas para seleção de regiões de interesse (ROI), construção de histogramas, entre outros. Também oferece suporte direto para o tratamento de imagens no formato médico digital (DICOM), recurso útil para este trabalho (THE MATHWORKS, 2008). Além de tais recursos, este software permite o desenvolvimento de aplicações em curto período de tempo uma vez que todos os recursos de máquina, tal como o uso de memória e conversão de tipos são gerenciados em background pelo núcleo do MATLAB. As desvantagens de seu uso são o custo de aquisição de licenças e o tipo de aplicação gerada, normalmente interpretada, requerendo uma instalação prévia do MATLAB para ser executada, portanto acabou não sendo utilizado.

Outro toolkit testado foi o ITK (2007). O ITK é um conjunto de bibliotecas de código fonte aberto, orientado a objeto, destinado ao processamento de imagens, mais especificamente segmentação e registro (IBÁÑEZ et al., 2005). Segmentação é o processo de identificação e classificação de dados encontrados em uma representação digital. Tipicamente esta representação é uma imagem adquirida de um instrumento médico, como uma CT ou uma MR. Registro é a tarefa de alinhar ou desenvolver correspondências entre dados. Por exemplo, no ambiente médico, uma varredura de CT pode ser alinhada com uma varredura de MR para combinar as informações contidas em ambos. O ITK, no entanto, não dispõe de interface gráfica (GUI) e nem de sistema de visualização, logo, para a visualização dos resultados obtidos é necessário um pacote de visualização gráfica como a OpenGL ou o VTK.

O VTK (2007) é um sistema de código fonte aberto, orientado a objeto, livremente disponível para computação gráfica 3D, processamento de imagens, e visualização (SCHROEDER; AVILA; HOFFMAN, 2000). Este toolkit foi adotado nessa pesquisa para a implementação computacional. No VTK é possível gerar aplicações independentes (stand-

alone), em diferentes plataformas e linguagens de programação. Oferece um bom número de recursos inerentes à visualização científica (VC) e à visualização volumétrica (VV), além de várias formas de organizar e manipular dados. Vários casos de sucesso também motivaram essa escolha, dentre os softwares desenvolvidos com o uso do VTK destacam-se o 3D-Slicer (2007), o ParaView (2007) e o InVesalius (2007). Bibliotecas gráficas como o VTK aumentam a qualidade e a eficiência do trabalho do pesquisador, a prova disso pode ser dada pela grande quantidade de trabalhos publicados envolvendo de forma direta ou indireta o uso dessa tecnologia, merecendo destaque os trabalhos de Moreno (2004), Poh e Kitney (2005), Müller Adaime (2005), Buriol (2006) e Meurer et al. (2008). Sendo que este último usa o software InVesalius para reconstruir a região maxilofacial. E finalmente, a grande quantidade de documentação disponível foi fator decisivo para a escolha do VTK como a biblioteca-gráfica base de desenvolvimento neste projeto. A empresa disponibiliza dois livros que cobrem grande parte dos recursos: Kitware (2006) e Schroeder, Martin e Lorensen (2006), e além da documentação on-line das classes e métodos (VTK DOCUMENTATION, 2007), existe também uma lista de discussão com muitos membros cadastrados onde é possível lançar dúvidas e discutir soluções.

Outra ferramenta pesquisada para desenvolvimento de aplicações de VC e VV foi a OpenVL (2007). Segundo Lakare e Kaufman (2003), trata-se de uma biblioteca modular, extensível e de alta performance para manipular conjuntos de dados volumétricos. A biblioteca fornece uma API (Application Programming Interface) padrão, uniforme e de fácil uso para acessar dados volumétricos, permitindo leitura/escrita de dados volumétricos de/para arquivos em diferentes formatos usando plugins. Fornece um framework para implementação de algoritmos como plugins que podem ser incorporados dentro das aplicações do usuário. Estes plugins são implementados como bibliotecas compartilhadas as quais podem ser dinamicamente carregadas quando necessário. A OpenVL é de código fonte aberto, e está livremente disponível na web. A desvantagem de seu uso é que os algoritmos de reconstrução e geração de malha para serem implementados requerem maior tempo de desenvolvimento.

Existe também um toolkit voltado especificamente para a área médica, o MITK (Medical Imaging Interaction Toolkit) – pacote para interação com imagens médicas. Segundo Wolf et al. (2004) o objetivo do MITK (2007) é facilitar a criação de softwares com base em imagens para uso clínico. MITK é uma biblioteca de classes que tem por base estender o ITK e o VTK. O ITK fornece algoritmos para registro e segmentação e forma a base da implementação, enquanto o VTK fornece a capacidade de visualização. MITK reusa virtualmente qualquer objeto do ITK e do VTK, então ele não compete com os dois, mas os

estende, combinando os recursos de ambos e adicionando novos, os quais são requeridos por softwares que precisam interagir com imagens médicas. O MITK é de código fonte aberto, no entanto, o seu uso foi desencorajado por não ter muita documentação disponível.

Síntese dos Toolkits de Visualização Científica Estudados

A Figura 2.6 apresenta de forma resumida as características dos pacotes de visualização científica pesquisados, com destaque para o VTK que foi o toolkit adotado para implementação neste trabalho.

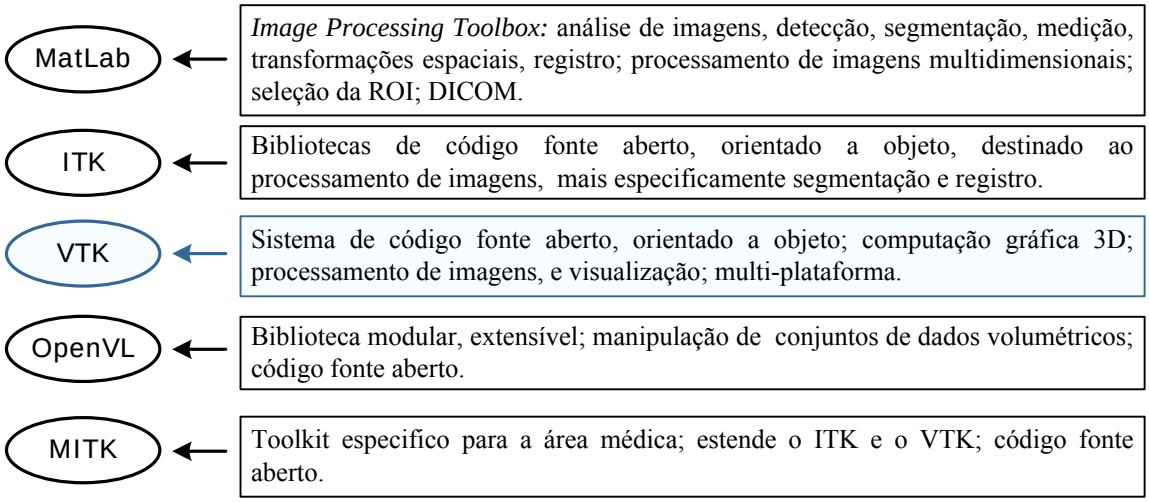


Figura 2.6 – Os toolkits de visualização científica estudados.
Fonte: O autor.

Outros sistemas foram encontrados durante a pesquisa, como o VolVis (2007), o AVS (2007) e o OpenDX (2007) formalmente conhecido como Data Explorer. Estes, embora investigados, não foram testados como os cinco primeiros (Figura 2.6).

3 TÉCNICAS DE RECONSTRUÇÃO E GERAÇÃO DE MALHA

No capítulo anterior foram citados vários métodos numérico-computacionais relacionados à reconstrução tridimensional e à geração de malha, dentro do que foi encontrado na bibliografia atual. O presente capítulo, no entanto, dedica-se exclusivamente aos métodos que foram utilizados na fase de implementação, um para a reconstrução de superfícies: o *Marching Cubes*, e o outro associado ao processo de geração de malha de volumes: a *Triangulação de Delaunay*.

Tais métodos foram selecionados com base no seguinte levantamento:

- Existência de casos de uso de sucesso e exemplos de sua aplicação;
- Implementação em toolkits e bibliotecas de desenvolvimento;
- Capacidade de tratar formas tridimensionais e geometrias irregulares;
- Aplicabilidade a modelos biomecânicos, neste caso, regiões ósseas.

As próximas seções abordam, isoladamente, cada um deles.

3.1 MARCHING CUBES

O *Marching Cubes* (MC) desenvolvido por Lorensen e Cline (1987) é um dos métodos mais conhecidos para a reconstrução 3D. Ele produz modelos de triângulos de isosuperfícies $F(x, y, z) = c$ (onde c=limiar) de uma dada função escalar sobre uma grade de cubos. Usualmente o método MC é considerado como o método básico para renderizar superfícies em aplicações médicas. Todavia, ele pode ser aplicado em muitas outras áreas, como por exemplo, a visualização de funções implícitas ou a visualização de resultados de cálculos com o Método dos Elementos Finitos.

A idéia do método Marching Cubes pode ser mais facilmente entendida pelo seu equivalente no plano, o *Marching Squares*. O fundamento básico desta técnica é que um contorno somente pode passar através de uma célula em um número finito de modos. Esta técnica trata as células independentemente. Uma tabela de casos é construída para enumerar todos os possíveis estados topológicos de uma célula, dando as combinações de valores escalares nos pontos das células. O número de estados topológicos depende do número de

vértices das células, e do número de relacionamentos interior/exterior que um vértice pode ter com respeito ao valor de contorno. Um vértice é considerado interior a um contorno se seu valor escalar é maior do que o valor escalar da linha de contorno. Vértices com valores escalares menores do que o valor de contorno são denominados exteriores ao contorno. Por exemplo, se uma célula tem quatro vértices e cada vértice pode ser interior ou exterior ao contorno, então existem $2^4 = 16$ possíveis modos do contorno passar através da célula. Na tabela de casos não se está interessado em que lugar o contorno passa através da célula (interseção geométrica), mas como ele passa através da célula (topologia do contorno na célula).

A Figura 3.1 mostra as 16 combinações possíveis para uma célula quadrada. Um índice dentro da tabela de casos pode ser computado para codificar o estado de cada vértice como um dígito binário. Para dados 2D representados em uma grade retangular, pode-se representar os 16 casos com índices de 4 bits. A partir do momento que o caso apropriado é selecionado, a localização da interseção da linha de contorno/aresta da célula pode ser calculada usando interpolação. O algoritmo processa uma célula e então move, ou marcha para a próxima célula. Depois que todas as células são visitadas, o contorno estará completo.

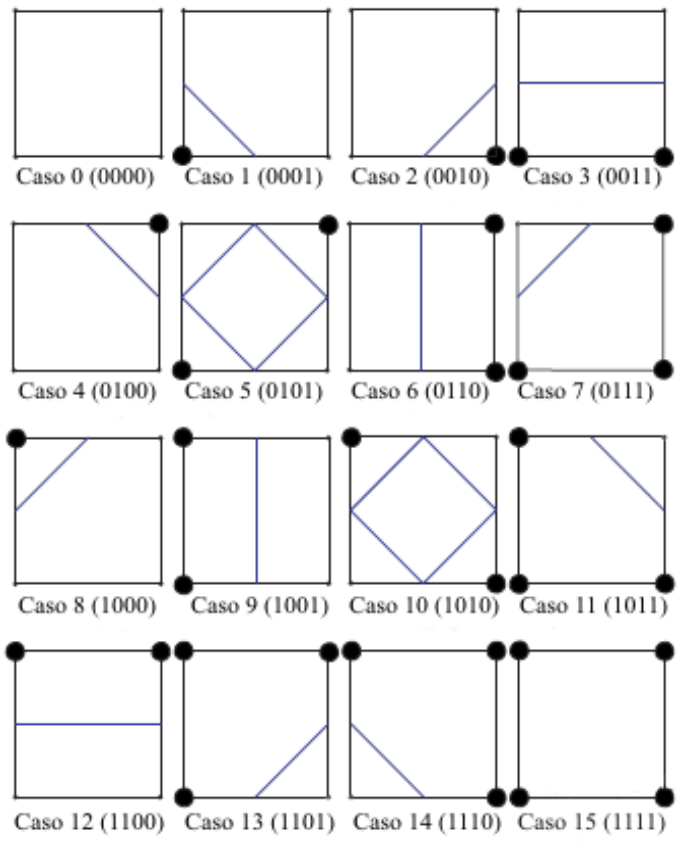


Figura 3.1 – Marching Squares
Fonte: Milne, Nicolls e Jager (2004).

Em resumo, o algoritmo “marchante” procede da seguinte forma:

1. Seleciona uma célula.
2. Calcula o estado interior/exterior para cada vértice da célula.
3. Cria um índice para o armazenamento do estado binário de cada vértice em um bit separado.
4. Usa o índice para olhar o estado topológico de cada célula na tabela de casos.
5. Calcula a localização do contorno (via interpolação) para cada aresta.

Este processo irá construir primitivas geométricas em cada vértice. Em células vizinhas, vértices e contornos duplicados podem ser criados. Esta duplicação pode ser eliminada pelo uso de uma operação especial de junção de pontos coincidentes. Note que a interpolação ao longo de cada aresta deve ser feita na mesma direção, senão o erro de arredondamento poderá gerar pontos que não sejam precisamente coincidentes, e não irá juntá-los apropriadamente.

Existem vantagens e desvantagens no uso dessa técnica. O algoritmo de Marching Squares é fácil de implementar, mas quando a técnica é estendida para três dimensões, no entanto, torna-se muito mais difícil.

Como mencionado previamente, o análogo 3D do Marching Squares é o Marching Cubes. Neste há 256 diferentes combinações de valores escalares, dado que existem oito pontos em uma célula cúbica, podendo estar em dois estados (interno/externo), ou seja, 2^8 combinações. A Figura 3.2 mostra essas combinações reduzidas a 15 casos usando o argumento da simetria. Usam-se combinações de rotação e espelhamento para produzir os demais casos topologicamente equivalentes.

Usando a mesma idéia do Marching Squares, o MC processa uma célula de cada vez. Ele determina como a isosuperfície intersecta uma dada célula, e então move para a próxima célula. Isto é feito numerando-se os vértices, de um a oito, e armazenando o resultado em um byte. Se o vértice possui um valor superior ao de referência, a posição do vértice no byte é “1”, caso contrário, “0” (Figura 3.3). Os bytes são posteriormente processados para se determinar a localização da configuração triangular correspondente.

As 256 configurações possíveis são incorporadas na tabela de casos. Cada entrada na tabela contém um padrão de triângulo para a correspondente configuração. Para o caso 0 não há triângulos, porque todos os oito vértices estão dentro ou fora da isosuperfície. Para o caso 1 o padrão consiste de um triângulo, a partir do qual a isosuperfície separa um vértice dos outros sete. Os padrões para as outras configurações têm dois, três ou quatro triângulos.

Assim, o tipo de cada célula na grade é determinado, e vértices reais, formados pela intersecção das isosuperfícies com as arestas, são substituídos no correspondente padrão de triângulos. Interpolações lineares ao longo das arestas são usadas para determinar as coordenadas dos vértices.

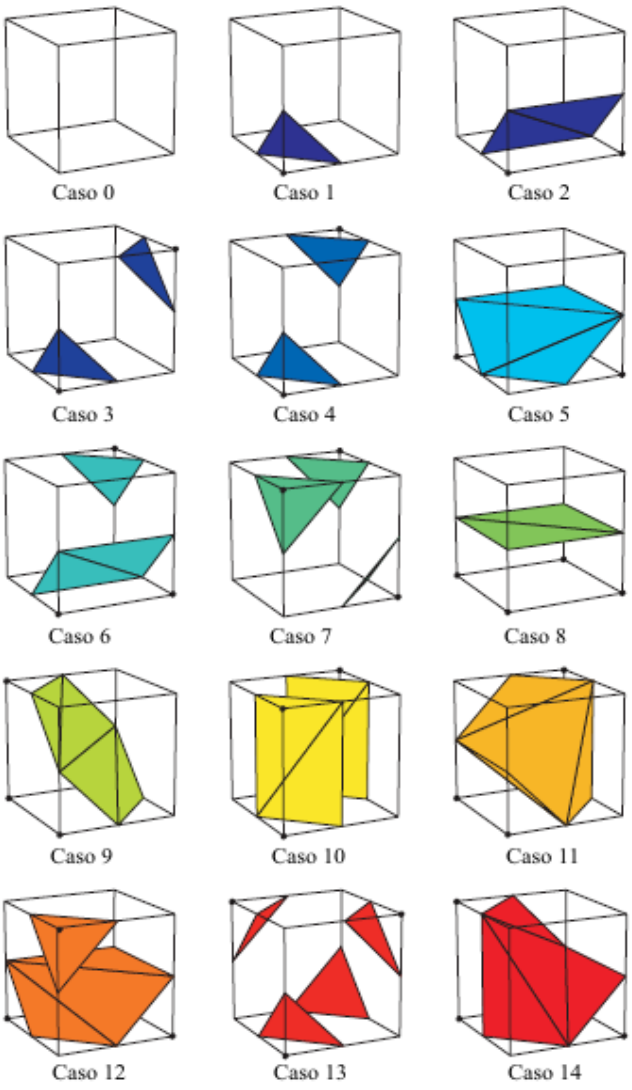


Figura 3.2 – Marching Cubes, casos 3D para geração de isosuperfícies.
Fonte: Milne, Nicolls e Jager (2004).

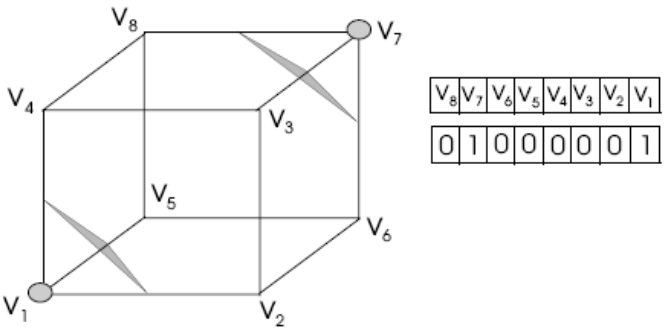


Figura 3.3 – Exemplo de um voxel com numeração dos vértices e respectiva representação em um byte.
Fonte: Lam, Loke e Buf (2001).

Ambigüidades no Algoritmo de Marching Cubes

Usando as 15 superfícies ilustradas na Figura 3.2 pode resultar em várias ambigüidades. Elas são facilmente visíveis na forma de buracos (Figura 3.4a).

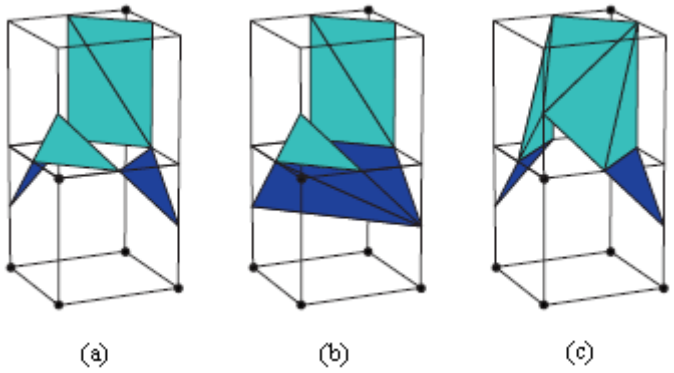


Figura 3.4 – Uma das ambigüidades do algoritmo de Marching Cubes.
Fonte: Milne, Nicolls e Jager (2004).

A Figura 3.4c mostra uma possível solução, baseada nos estudos de Chernyaev (1995), que para tentar resolver o problema apresentou um estudo estendendo o número de configurações básicas de 15 para 33, então uma nova tabela com 33 superfícies foi criada, conforme pode ser observado pela Figura 3.5.

Outra técnica utilizada para tentar remediar o problema será rapidamente comentada. É aquela que substitui os cubos por tetraedros, a técnica de *Marching Tetrahedra* (MT). Este algoritmo não exibe casos de ambigüidade, mas infelizmente o MT gera isosuperfícies consistindo de mais triângulos, e a construção requer uma escolha a respeito da orientação do tetraedro. Esta escolha pode causar impacto na isosuperfície por causa da interpolação ao longo da face diagonal. Em 2D a diagonal pode ser escolhida arbitrariamente, mas em 3D a diagonal é restrita à célula vizinha.

Além dos tipos regulares como quadrados e cubos, Marching Cubes pode ser aplicado a qualquer tipo de célula topologicamente equivalente a um cubo (isto é, hexaedro ou voxel não-cúbico).

Segundo Schroeder, Martin e Lorensen (2006), no VTK, biblioteca gráfica adotada para a fase de desenvolvimento, a técnica geral de Marching Squares e Marching Cubes é estendida a outros tipos topológicos, como marcha de linhas, triângulos, e tetraedros, a fim de contornar células desses tipos (ou composição de células formadas por esses tipos).

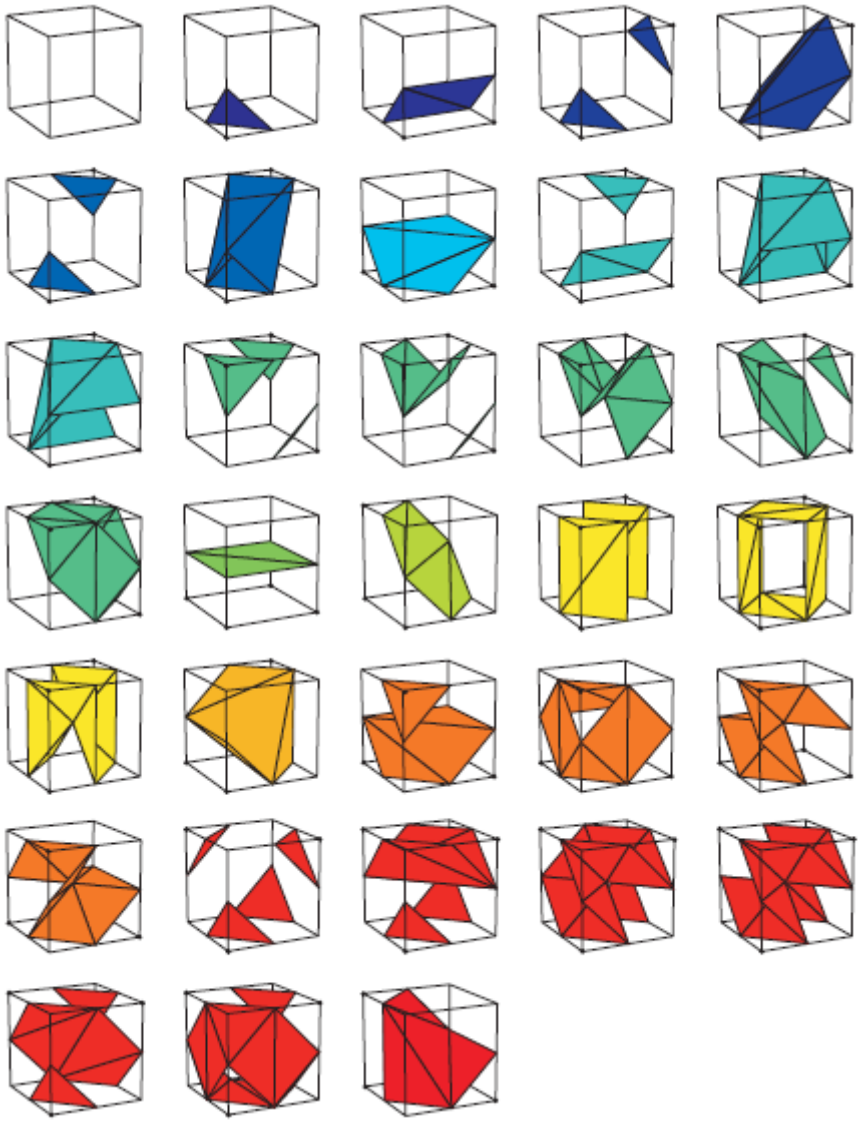


Figura 3.5 – As 33 superfícies estendidas de Marching Cubes, por Chernyaev.
Fonte: Milne, Nicolls e Jager (2004).

3.2 TRIANGULAÇÃO DE DELAUNAY

Técnicas de triangulação permitem construir a topologia diretamente de pontos não organizados. Os pontos são então triangulados para criar uma estrutura topológica consistindo de n-dimensional simplexos que contornam completamente o conjunto de pontos e de combinações lineares dos pontos, conhecido como *convex hull*. O resultado da triangulação é um conjunto de triângulos (2D) ou tetraedros (3D), dependendo da dimensão do conjunto de pontos de entrada.

Definição

Uma triangulação n-dimensional de um conjunto de pontos $P = \{p_1, p_2, p_3, ..., p_n\}$ é uma coleção de n-dimensional simplexos cuja definição de pontos encontra-se em P. Os simplexos não intersectam um ao outro e compartilham elementos de fronteira como arestas ou faces.

A triangulação de Delaunay é um caso particular. Ela tem a propriedade de que a circunferência circunscrita de qualquer n-dimensional simplexo não contém outros pontos de P exceto os n+1 pontos definidos do simplexo (Figura 3.6a).

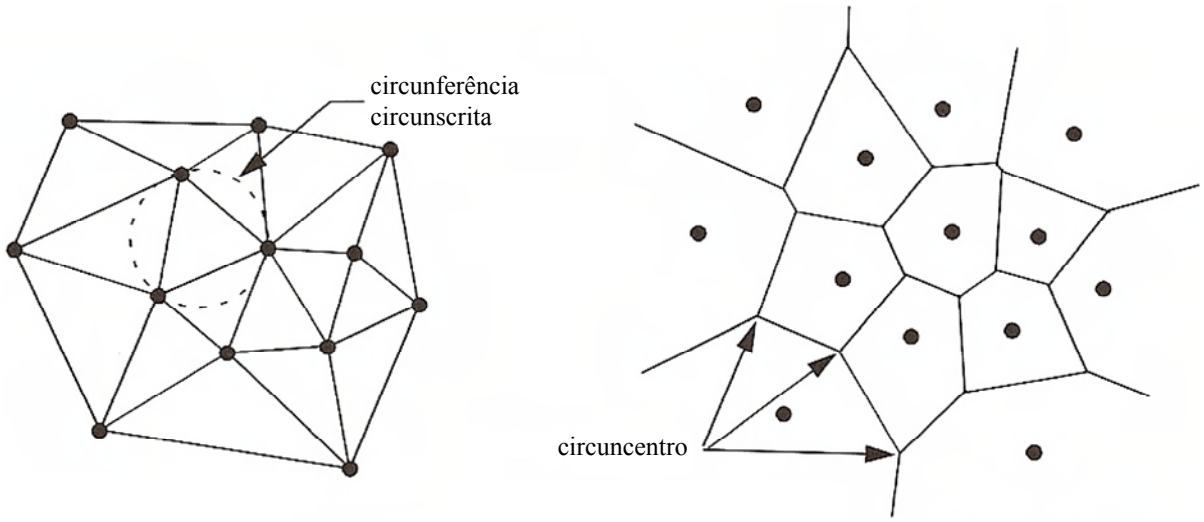


Figura 3.6 – (a) Triangulação de Delaunay; (b) Construção de Dirichlet.
Fonte: Schroeder, Martin e Lorensen (2006).

Resumidamente, pode-se dizer que a triangulação de Delaunay, no plano:

- encontra segmentos de reta que conectam um conjunto de pontos;
- os segmentos particionam o conjunto de pontos em triângulos;
- nenhum desses segmentos pode cruzar com nenhum outro;
- cada ponto é vértice de pelo menos um triângulo.

As propriedades da triangulação de Delaunay estão intimamente ligadas à construção de Dirichlet (também conhecida como diagrama de Voronoi), a qual é considerada a forma geométrica dual da triangulação de Delaunay (Figura 3.6b). A construção de Dirichlet é a região do espaço onde cada divisão representa o espaço fechado para um ponto p_i (estas divisões são as células de Voronoi). Uma triangulação de Delaunay pode ser feita a partir da

construção de Dirichlet pela criação de arestas entre células de Voronoi que compartilham n–1 contornos comuns. Os vértices da construção de Dirichlet estão localizados nos circuncentros dos círculos mostrados na triangulação de Delaunay (SCHROEDER; MARTIN; LORENSEN, 2006).

Uma propriedade interessante da triangulação de Delaunay é que existe uma única triangulação que maximiza a soma dos menores ângulos de cada triângulo da malha. Em outros termos, dada uma nuvem de pontos, a triangulação de Delaunay é a que resulta em um conjunto de triângulos o mais próximo possível de triângulos eqüiláteros. Isto mostra que a triangulação de Delaunay em duas dimensões é ótima, enquanto que em 3D isto nem sempre ocorre, e será discutido a seguir.

A triangulação de Delaunay pode ser utilizada como algoritmo de geração de malhas triangulares ou tetraedrais. Mas criar malhas tetraedrais de boa qualidade é um problema de difícil solução. A dificuldade da tarefa ocorre por diversos motivos. Primeiro porque o simples tamanho do resultado das malhas para armazenamento computacional requer estruturas de dados robustas e organizadas. Há também a dificuldade matemática básica que faz a malha de tetraedros significativamente mais difícil que a sua contraparte plana, isto porque os tetraedros gerados nem sempre são regulares, não ocupando todo o espaço 3D, enquanto que os triângulos conseguem preencher todo o plano. Diferente dos casos 2D, cada vértice bem espaçado pode criar elementos 3D degenerados tal como slivers (fatias finas) (ALLIEZ et al. 2005). A Figura 3.7 a seguir mostra alguns elementos tetraédricos que podem ser gerados pela triangulação de Delaunay em três dimensões.

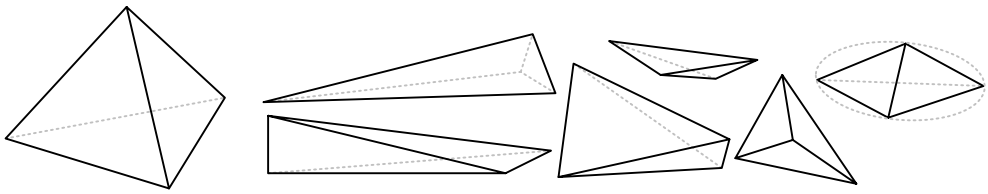


Figura 3.7 – Tipos de tetraedros que podem ser gerados com a triangulação de Delaunay em três dimensões. Mais à esquerda um tetraedro regular (ideal), e mais à direita um sliver (fatia fina).
Fonte: Alliez et al. (2005).

Malhas com elementos bem proporcionais irão dar bons resultados para a análise. O ideal seria que os triângulos que compõem as faces dos tetraedros fossem todos eqüiláteros, no entanto é comum aparecer casos degenerados como os slivers. Slivers ocorrem quando os quatro vértices do tetraedro ocupam um grande círculo da esfera circunscrita e estão igualmente distribuídos ao longo desse círculo (Figura 3.7).

Para medir a qualidade dos elementos gerados costuma-se levar em conta os aspectos de proporcionalidade (aspect ratio) que é a razão entre o raio da esfera circunscrita para o raio da esfera inscrita. Quanto menor for o aspecto de proporcionalidade melhor será o tetraedro.

Outro problema comum em três dimensões são os casos degenerados que ocorrem quando mais do que quatro pontos encontram-se na fronteira da esfera circunscrita, representando diferentes escolhas. Quando isto acontece, uma decisão inconsistente pode ser tomada aceitando ou rejeitando um dado tetraedro. Um dos métodos existentes para resolver este problema é perturbar levemente as coordenadas de um ponto de uma nova entrada sempre que o ponto encontrar-se ambiguamente na esfera circunscrita. A perturbação deve ser pequena o suficiente, mas de modo a permitir a correta decisão a ser tomada em relação ao interior/exterior durante o cálculo de precisão finita. Depois de completar a triangulação, todos os nós perturbados são restaurados para sua posição original (KANAGANATHAN; GOLDSTEIN, 1991). Outras soluções existem, por exemplo, Schroeder, Geveci e Malaterre (2004) resolvem o problema através de operações de ordenação dos simplexos.

Quanto à implementação da triangulação de Delaunay, diferentes algoritmos foram criados nestas últimas décadas, inclusive com vários refinamentos e restrições de contorno. Na documentação do VTK são citados os algoritmos de Bowyer (1981) e Watson (1981) que trabalham com inserção incremental de pontos. Um diagrama esquemático desse tipo de algoritmo em duas dimensões pode ser observado pela Figura 3.8, a seguir.

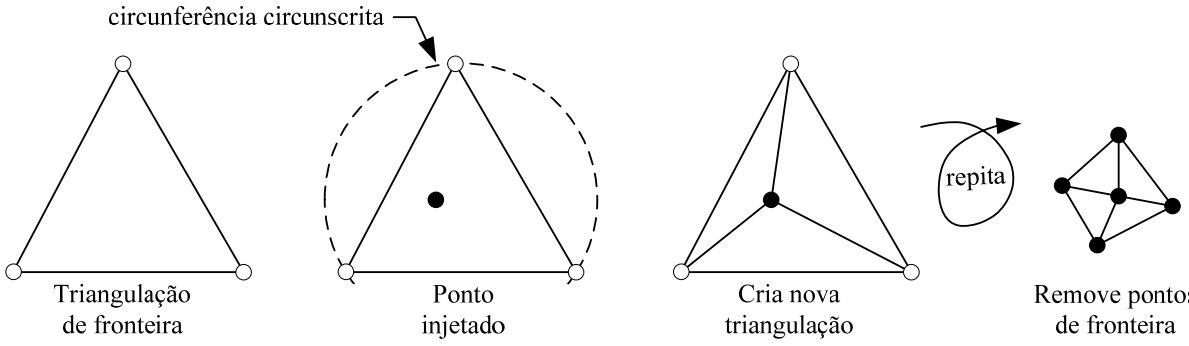


Figura 3.8 – Computação da triangulação de Delaunay com o uso da técnica de inserção incremental de pontos. Fonte: Schroeder, Martin e Lorensen (2006).

Para três dimensões Kanaganathan e Goldstein (1991) fizeram uma comparação entre quatro algoritmos existentes para geração de malha tetraédrica, comparando a sua complexidade e a qualidade dos elementos produzidos. Em seu trabalho o método de Watson se mostrou melhor em termos de qualidade da malha gerada. Ele sintetizou a idéia do algoritmo de Watson da seguinte forma:

a. Inicia com um tetraedro que contém todos os pontos a serem adicionados; novos tetraedros internos são formados quando novos pontos são incluídos, um de cada vez.
b. Em um típico estágio do processo, um novo ponto é testado para determinar qual esfera circunscrita contém o ponto. O tetraedro associado é removido, deixando um poliedro de inserção contendo o novo ponto.
c. Arestas conectando o novo ponto a todas as faces triangulares da superfície do poliedro de inserção são criadas, definido um tetraedro que preenche o poliedro de inserção. Combinando estes com o tetraedro do lado de fora do poliedro de inserção produz uma nova triangulação que contém o novo ponto adicionado.

Triangulação de Delaunay com Restrição dos Contornos no Plano

Certos autores têm estudado o problema que consiste em: "*dado um conjunto de pontos obter a triangulação de Delaunay (e seu dual, o diagrama de Voronoi)*". No entanto, formulações como: "*dado um conjunto de pontos e um conjunto de arestas entre esses pontos que não se cortam, obter a triangulação de Delaunay que inclui estas arestas*" surgiram na busca da solução de problemas correlatos. Para resolver tais situações alguns pesquisadores implementaram restrições no algoritmo inicial de Delaunay e estas mudanças ficaram conhecidas em geometria computacional como CDT (constrained Delaunay triangulation) ou triangulação de Delaunay restrita. Por exemplo, Anglada (1997) implementou mudanças no algoritmo que permitiu triangular polígonos com concavidades e com possíveis buracos. Para isto ele criou dois novos procedimentos, um para a inserção de novos vértices na triangulação baseado em um algoritmo incremental, de modo semelhante ao que foi discutido anteriormente (Figura 3.8), e outro para a inserção de arestas.

O algoritmo para a inserção de uma aresta *ab* em uma CDT de um gráfico em duas dimensões, formulado por Anglada, segue os seguintes passos:

a. Remove os triângulos $t_1,...,t_k$ cortados por *ab* da CDT, então uma região sem triangulação é deixada.
b. Adiciona a aresta *ab* ao resultado.
c. Re-triangula as regiões acima e abaixo da aresta *ab* que não foram trianguladas no primeiro passo.

A idéia do algoritmo é simples e pode ser melhor entendida pela Figura 3.9, abaixo:

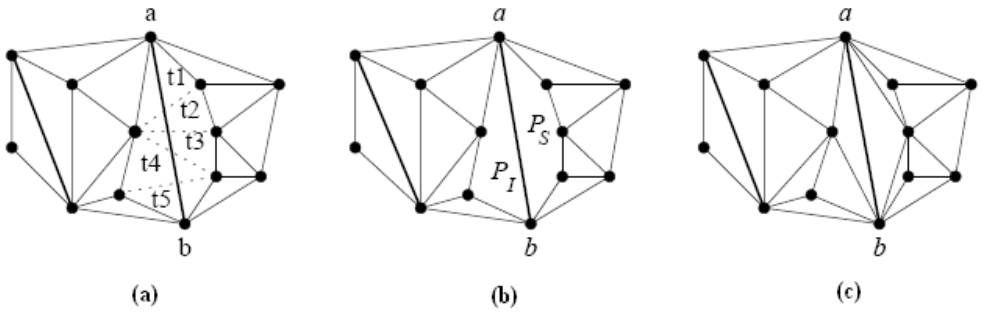


Figura 3.9 – Inserção de aresta com CDT: (a) localização e eliminação dos triângulos cortados pela aresta; (b) inserção da aresta; (c) re-triangulação das regiões adjacentes à aresta. / Fonte: Anglada (1997).

A Figura 3.10 mostra uma superfície com várias restrições triangulada com o algoritmo de Anglada.

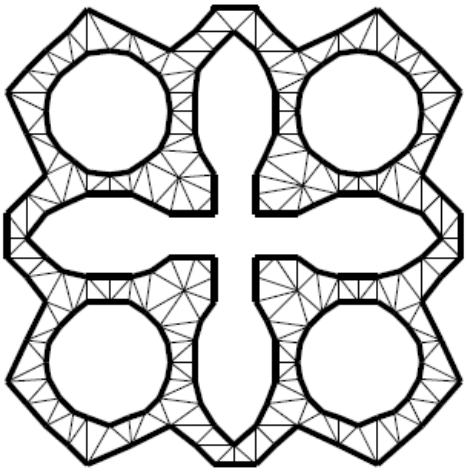


Figura 3.10 – Um exemplo da aplicação do algoritmo de Anglada
Fonte: Anglada (1997).

Refinamento da Triangulação de Delaunay em Três Dimensões

O algoritmo de Anglada resolve com eficiência vários problemas de triangulação no plano. Em relação ao espaço tridimensional também existem vários refinamentos atualmente implementados. Dentre os pesquisados, vale citar os de George, Hecht e Saltel (1990), o de Li (2000), Alliez et al. (2005) e Kolingerová e Žalic (2006). A seguir será apresentado o algoritmo de Alliez por ser um dos mais atuais e por preocupar-se com a qualidade da malha gerada.

O algoritmo de Alliez é mais complexo que os outros por usar o princípio de variação de energia e, segundo o autor, pode ser assim sintetizado:

Lê a malha de fronteira $\partial\Omega$ como entrada
Organiza a estrutura de dados & realiza o pré-processamento
Calcula o tamanho do campo μ
Gera posições iniciais x_i dentro de Ω
Faz
    Constrói a triangulação de Delaunay ($\{x_i\}$)
    Move posições x_i para sua posição ótima x_i^*
Até (convergência ou critério de parada)
Extrai interior da malha

O algoritmo toma como entrada uma malha de superfície de triângulos fechada, definindo a fronteira do domínio denominado $\partial\Omega$. A máxima do algoritmo se dá na fase de minimização da energia, alternando a conectividade e otimizando a geometria. Para uma dada conectividade, a energia é minimizada movendo cada vértice interior x_i para sua posição ótima. Para produzir a malha final é usada a triangulação de Delaunay que gera a malha de tetraedros sobre a *hull* convexa conforme pode ser observado pela Figura 3.11a.

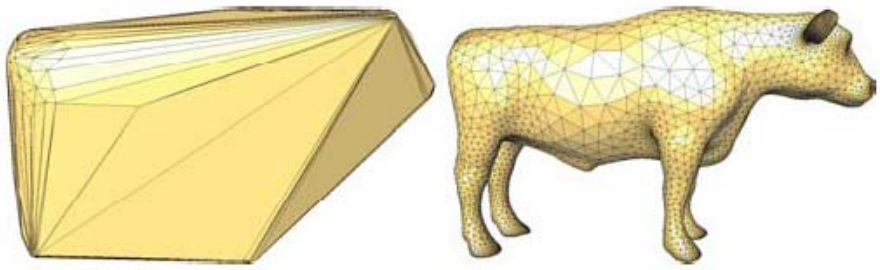


Figura 3.11 – Extração final da malha: (a) a triangulação de Delaunay cobre a hull convexa; (b) o último estágio do algoritmo extrai os tetraedros internos. / Fonte: Alliez et al. (2005).

Depois disso ainda é necessário extrair os tetraedros internos e isto é feito tomando parte do domínio de entrada Ω e computando a proporção entre a distância *d* do centro da esfera circunscrita até a fronteira inicial $\partial\Omega$ e o raio da esfera; se a taxa é menor que o limiar (threshold) definido (0,4 segundo o experimento do autor) então o tetraedro é considerado interno e irá compor a malha final (Figura 3.11b).

Muitos passos foram omitidos por questão de simplicidade uma vez que o objetivo final aqui é mostrar os recursos e a potencialidade do algoritmo. A Figura 3.12 mostra a homogeneidade e a qualidade da malha gerada. O detalhamento, bem como as fases e os teoremas de minimização da energia variacional podem ser encontrados em Alliez et al. (2005).

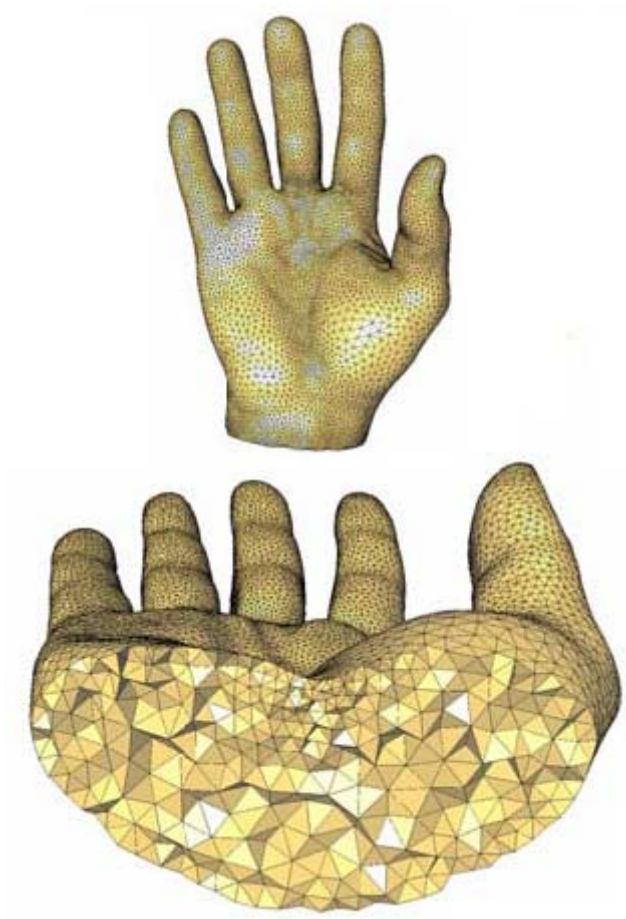


Figura 3.12 – Exemplo da qualidade da malha tetraédrica gerada com o algoritmo de Alliez.
Fonte: Alliez et al. (2005).

As várias modificações implementadas no algoritmo de Delaunay ultimamente, mostram que esta técnica ainda é bastante utilizada no processo de geração de malhas tetraedrais.

O próximo capítulo irá abordar os recursos computacionais pertinentes ao toolkit de visualização científica utilizado neste projeto.

4 O VTK (THE VISUALIZATION TOOLKIT)

Este capítulo não tem a pretensão de esgotar o assunto sobre o VTK, mas apresentar os conceitos fundamentais, necessários para a utilização desse importante toolkit no desenvolvimento de sistemas de visualização científica, como o bioMeshCreate. Os conceitos apresentados a seguir, abordam de maneira generalizada a sua estrutura, organização e funcionamento, de modo a fornecer um panorama geral sobre a biblioteca gráfica. Estas informações podem ser encontradas, de forma expandida em Schroeder, Martin e Lorensen (2006), principais criadores da tecnologia. Este estudo constitui, portanto, os alicerces do desenvolvimento de aplicações com VTK.

4.1 O QUE É VTK

O Visualization ToolKit (VTK) é um sistema livremente disponível, de código fonte aberto (open-source), para computação gráfica 3D, processamento de imagens, e visualização.

Ele fornece uma variedade de formas de representação de dados, incluindo conjunto de pontos não-estruturados, dados poligonais, imagens, volumes, e malhas do tipo estruturada, retilínea e não-estruturada. Possui classes para leitura/importação e escrita/exportação de dados, a fim de possibilitar a comunicação com outros aplicativos. Conta com centenas de filtros para operar nesses dados, desde uma convolução da imagem até a triangulação de Delaunay. O modelo de renderização do VTK suporta ainda gráficos 2D, poligonal, volumétricos e aplicação de texturas, podendo ser usados em qualquer combinação (SCHROEDER; AVILA; HOFFMAN, 2000).

O projeto e implementação da biblioteca foram influenciados fortemente pelos princípios da programação orientada a objetos (OOP - Object Oriented Programming). Na época em que este projeto começou a ser desenvolvido a versão corrente era a 5.0.3.

É multi-plataforma, isto é, pode ser utilizado em mais de um sistema operacional, já tendo sido instalado e testado em plataformas baseadas em Unix, Linux, Windows e Mac OS. Para entender como o VTK alcança essa independência de plataforma é preciso entender sua organização, sua arquitetura.

4.2 ARQUITETURA

O toolkit consiste de dois subsistemas básicos: uma biblioteca compilada em C++, e várias interfaces interpretadas estendidas em camadas incluindo Tcl/Tk, Java, e Python, ou seja, um código-fonte, várias linguagens de programação.

A vantagem desse tipo de arquitetura é que é possível manter o núcleo compilado em C++ e construir a interface gráfica do usuário (GUI - Graphic User Interface) em cima de linguagens interpretadas, isolando desse modo, parte da complexidade. Mas havendo proficiência em C++ e a ferramenta adequada para fazê-lo, as aplicações podem ser construídas inteiramente em C++, obviamente mais eficientes, pois a aplicação passa a ser 100% compilada. A Figura 4.1 ilustra essa arquitetura.

Quanto à camada compilada, conta com mais de 700 classes em C++, cerca de 350,000+ linhas de código, sendo modelada e desenhada segundo Rumbaugh et al. (1994).

Deve-se destacar também que para o desenvolvimento com Tcl/Tk o processo é mais direto, pois já está estaticamente linkado ao build do vtk.exe. Para desenvolvimento com Python ou Java, ou para sistemas Linux ou Macintosh, é necessário compilar o VTK a partir do código fonte usando CMake e um sistema nativo para a plataforma em questão. A próxima seção se atém a este processo.

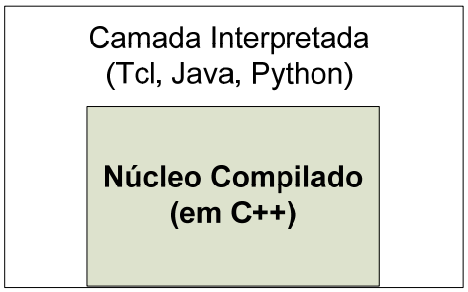


Figura 4.1 – Esquema da arquitetura do VTK
Fonte: Kitware (2006).

4.3 O GERADOR DE CÓDIGO CMAKE

O CMake é um gerador de código multi-plataforma. É usado para controlar o processo de compilação de software usando uma plataforma simples e compilador independente de sistema de arquivos. Gera código nativo, arquivos e workspaces que podem

ser usados no ambiente do compilador escolhido. Suporta ambientes complexos que requerem configurações de sistemas, geração de pré-processador e geração de código. O código fonte e a versão compilada podem ser encontrados livremente em CMake (2007).

Resumindo, ele é utilizado para produzir arquivos nativos para um determinado sistema operacional e compilador, de acordo com a sua disponibilidade. A versão do CMake utilizada neste trabalho foi a 2.4 – patch 5. Nesta versão o gerador de código possibilitava a criação da camada interpretada do VTK para as linguagens Tcl, Java e Python, além de poder converter o código C++ standard para os ambientes do Microsoft Visual C++ 6.0, Microsoft Visual C++ 2003, MS Visual C++ 2005 e Borland C++ Builder 6.0.

Embora o CMake apresente recursos avançados de geração de código, a utilização dos seus recursos básicos é intuitiva. No sistema operacional Windows sua interface se parece com a Figura 4.2, abaixo.

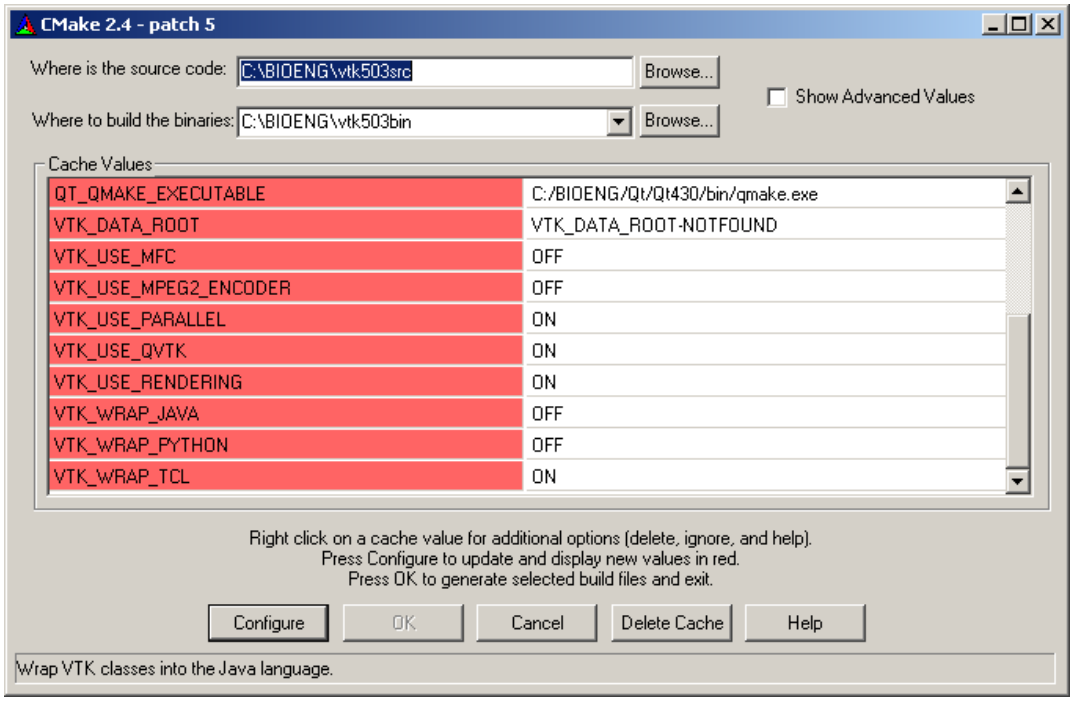


Figura 4.2 – Interface gráfica do programa CMake 2.4 sob a plataforma Windows.
Fonte: O autor.

É necessário, basicamente, configurar os caminhos de origem (código fonte do VTK) e destino (código compilado ou binário), além de alguns atributos que discriminam a linguagem, o compilador e os tipos de recursos que serão utilizados no desenvolvimento do projeto. Após isso, basta clicar sobre o botão configurar e, posteriormente em OK. Se tudo correr bem, o novo código será gerado no caminho previamente especificado e o CMake será automaticamente fechado ao final do processo. Se no decorrer do processo de

desenvolvimento, futuramente, for detectado a falta de algum recurso, como por exemplo, o processamento em paralelo, é possível executar novamente o CMake, acrescentando o recurso e re-gerando o código sem qualquer problema.

Para a implementação computacional subjacente a este trabalho foi feita a geração do código nativo para a plataforma MS Windows 32-bits e o compilador utilizado foi o Microsoft Visual C++ 2005.

Um último ponto sobre o CMake é sobre sua instalação. Deve-se baixar a versão compatível com o sistema operacional, e o compilador escolhido deve estar previamente instalado no equipamento.

Esta seção abordou resumidamente o gerador de código CMake, necessário para o desenvolvimento de aplicações com VTK. Informações mais detalhadas sobre sua instalação e uso podem ser encontradas em Martin e Hoffman (2006) e Kitware (2006) ou no próprio site do produto.

4.4 O MODELO DE OBJETOS

O modelo de objetos do VTK apresenta duas partes distintas. A primeira é o modelo gráfico, o qual é um modelo abstrato para gráficos 3D. A segunda é o modelo de visualização, que representa o modelo de fluxo de dados do processo de visualização. Existe ainda o modelo de execução, que é na verdade a seqüência com que o VTK processa as informações geradas pelos dois anteriores.

A compreensão desses modelos é essencial para o desenvolvimento de aplicações com VTK. A seguir será feito uma breve explanação sobre cada um deles.

4.4.1 O Modelo Gráfico

O modelo gráfico captura o aspecto essencial do sistema gráfico 3D em uma forma fácil de entender e usar. A abstração é baseada na indústria cinematográfica, com algumas influências da atual interface gráfica do usuário (GUI) dos sistemas baseados em janelas. No Visualization Toolkit existem sete objetos básicos que são usados para renderizar uma cena.

Existem muito mais objetos gráficos por trás da cena, mas estes sete são os mais frequentemente usados. Um diagrama ilustrativo desses objetos pode ser observado na Figura 4.3.

Segundo Schroeder, Martin e Lorensen (2006), estes objetos são:

a) *vtkRenderWindow* – gerencia a janela no dispositivo de exibição; um ou mais renderizadores desenham dentro de uma instância do vtkRenderWindow;
b) *vtkRenderer* – coordena o processo de renderização envolvendo luzes, câmeras, e atores;
c) *vtkLight* – constitui uma fonte de luz para iluminar a cena;
d) *vtkCamera* – define a posição de visão, ponto focal, e outras propriedades de visualização da cena;
e) *vtkActor* – representa um objeto renderizado na cena, incluindo suas propriedades e posição no sistema de coordenadas. (Nota: vtkActor é uma subclasse de vtkProp, que por sua vez, é uma forma mais geral de representar um ator, o qual inclui anotações e classes de desenho 2D);
f) *vtkProperty* – define as propriedades relacionadas à aparência de um ator, incluindo cor, transparência, e propriedades de iluminação tais como reflexão especular e difusão. Também representam propriedades como superfície sólida (cone da Figura 4.3) ou linhas (esfera da Figura 4.3);
g) *vtkMapper* – é a representação geométrica para um ator. Mais do que um ator pode referir-se ao mesmo mapper.

A classe vtkRenderWindow reúne todo o processo de renderização. Ela é responsável pelo gerenciamento da janela na tela do computador. Para PCs rodando Windows, ela será uma janela do Microsoft Windows, para sistemas Linux e UNIX ela será uma X Window e para Mac (OSX) uma janela Quartz. No VTK, instâncias de vtkRenderWindow são independentes de dispositivo. Isto significa que não é necessário se preocupar com o hardware em que o VTK está instalado, pois o software irá criar a instância adequada de vtkRenderWindow. A Figura 4.4 ilustra este isolamento.

Esta funcionalidade é atingida graças ao modelo em camadas do VTK. Para entender melhor, o modelo em camadas proporciona uma divisão dos módulos do sistema em grupos de acordo com sua tarefa, onde cada camada acrescenta um nível de abstração sobre a camada inferior, facilitando assim a manutenção do software, bem como a divisão das tarefas. Cada camada pode ser composta por vários componentes que interagem entre si ou com os

componentes da camada seguinte. As requisições são movidas do nível mais alto para o nível mais baixo. As respostas para as requisições ou notificações de eventos trafegam no sentido oposto (BUSCHMANN et al. 1996).

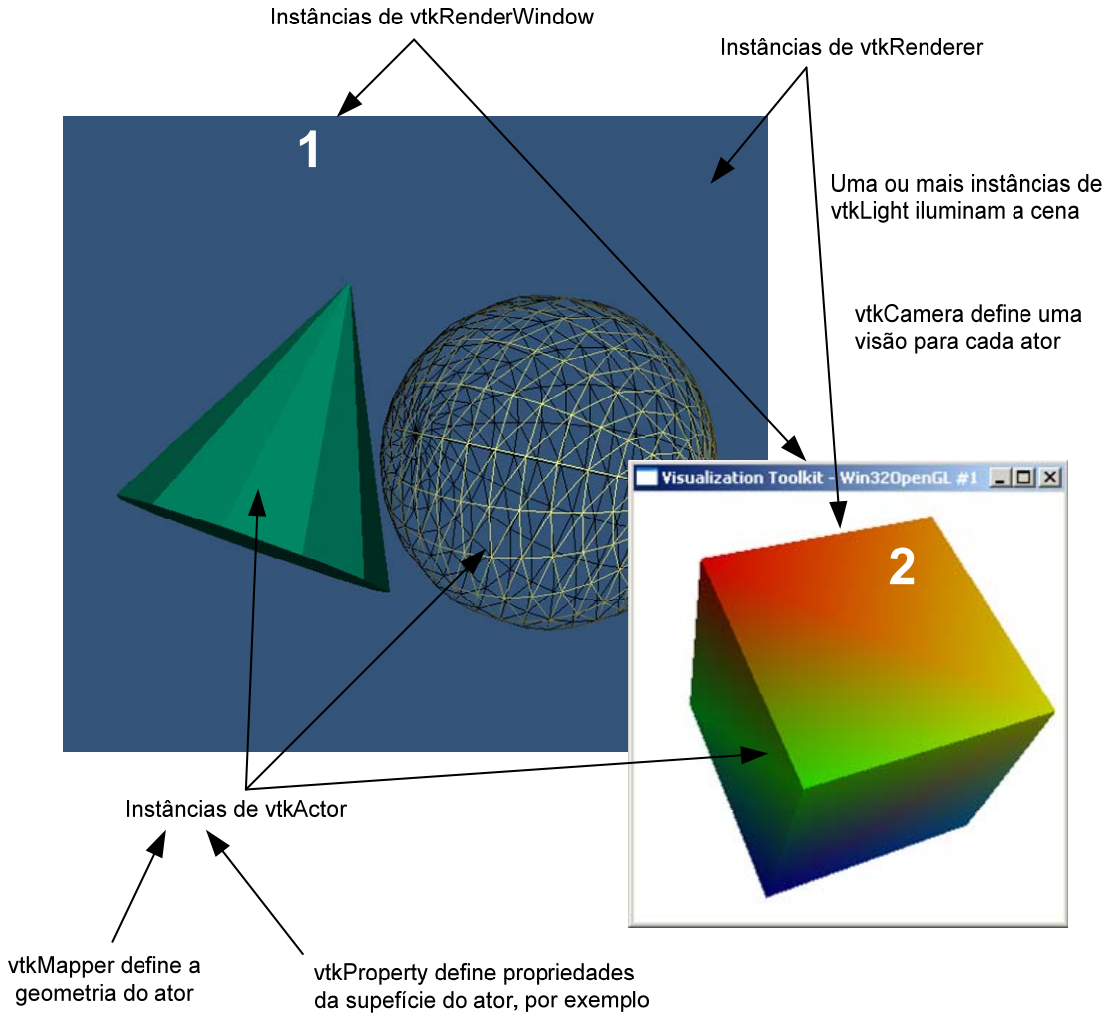


Figura 4.3 – Diagrama ilustrativo do modelo gráfico do vtk. A primeira janela de renderização foi obtida com bioMeshCreate, já a segunda através da execução de um exemplo que acompanha a documentação do vtk.
Fonte: Adaptado de Schroeder, Martin e Lorensen (2006).

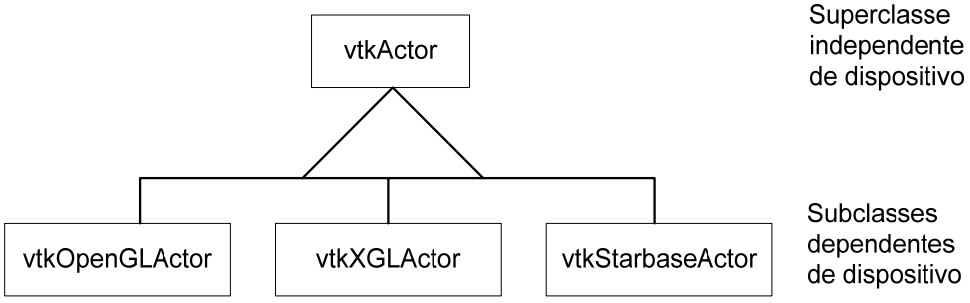


Figura 4.4 – Diagrama de classes do vtk: independência de dispositivos gráficos.
Fonte: Schroeder, Martin e Lorensen (2006).

A classe vtkRenderer é a responsável pela coordenação das luzes, câmera, e atores para produzir uma imagem. Cada instância mantém uma lista de atores, luzes e uma câmera ativa para uma cena em particular. Ao menos um ator deve ser definido, mas se as luzes e a câmera não são definidas, elas serão criadas automaticamente pelo renderizador. Em tais casos os atores são centralizados na imagem e a visualização padrão da câmera se dá sob o eixo z. Instâncias da classe vtkRenderer também fornecem métodos específicos para cor de fundo e iluminação do ambiente. Métodos também são avaliados para converter, visualizar e exibir sistemas de coordenadas.

Instâncias da classe vtkLight iluminam a cena. Várias variáveis de instância para orientar e posicionar as luzes estão avaliadas. Isto possibilita habilitar/desabilitar, bem como definir suas cores. Normalmente ao menos uma luz está ativa para iluminar a cena. Se nenhuma luz é definida e ativada, o renderizador constrói uma luz automaticamente. Luzes no VTK podem ser do tipo posicional ou infinita. Luz posicional possui ângulo cônico e fator de atenuação. Luz infinita projeta os raios paralelos um ao outro.

Câmeras são construídas pela classe vtkCamera. Importantes parâmetros incluem posição da câmera, ponto focal, localização ou planos de corte frontal e dorsal, direção vetorial de visão, e campo de visão. Câmeras também possuem métodos para simplificar sua manipulação, e isto inclui elevação, azimute, zoom e rolagem. Similar ao vtkLight, uma instância de vtkCamera será criada automaticamente pelo renderizador se nenhuma for definida.

Instâncias da classe vtkActor representam objetos na cena. Em particular, vtkActor combina propriedades dos objetos (cor, sombra, etc.), definição geométrica, e orientação no sistema global de coordenadas. Este é implementado por trás da cena mantendo-se variáveis de instância que se referem a instâncias de vtkProperty, vtkMapper, e vtkTransform. Normalmente não é necessário criar propriedades ou transformações explicitamente, pois elas são automaticamente criadas e manipuladas usando os métodos do vtkActor. Mas é necessário criar uma instância do vtkMapper (ou uma de suas subclasses). O mapper vincula os dados do processo de visualização (pipeline de visualização) para o dispositivo gráfico e será visto na seqüência.

Instâncias da classe vtkProperty afetam a aparência do ator renderizado. Quando os atores são criados, uma instância de suas propriedades é automaticamente criada com ele, mas também é possível criar propriedades dos objetos diretamente e então associá-las com um ou mais atores. Este modo é interessante, pois os atores podem compartilhar propriedades comuns. Finalmente, vtkMapper (e suas subclasses) definem a geometria dos objetos e,

opcionalmente, cor dos vértices. Em adição, vtkMapper pode referir-se a uma tabela de cores (vtkLookupTable) que são usadas para aplicar cor à geometria. Mas por hora, o mais importante é entender que o vtkMapper é um objeto que representa a geometria e os demais tipos de dados de visualização.

Há ainda outro objeto que merece destaque, embora não mencionado na lista anterior, é o vtkRenderWindowInteractor, que captura eventos (tais como os cliques do mouse e seu movimento) para um renderizador na janela de renderização. Esta classe captura eventos do mouse, e outros associados à movimentação da câmera, rotação, seleção de um ator e assim por diante. Instâncias dessa classe são associadas com a janela de renderização através do método setRenderWindow().

4.4.2 O Modelo de Visualização

O papel do modelo gráfico é transformar dados gráficos em figuras. O papel do modelo de visualização é transformar informações em dados gráficos. Um outro modo de entender isto é que o modelo de visualização é responsável por construir a representação geométrica que é então renderizada pelo modelo gráfico.

O Visualization Toolkit está baseado no paradigma de fluxo de dados adotado por muitos sistemas comerciais. Neste paradigma, módulos são conectados em uma rede ou cadeia de execução (network), também denotada como pipeline de execução. Os módulos executam operações algorítmicas nos dados conforme o fluxo através dessa rede. A execução na rede ou cadeia de visualização é então controlada em resposta à demanda por dados (demand-driven) ou em resposta às entradas ou solicitações do usuário, também conhecidas no meio computacional como resposta à eventos (event-driven). O grande atrativo deste modelo é sua flexibilidade, pois pode ser rapidamente adaptado a diferentes tipos de dados ou novas implementações nos algoritmos.

Para transformar as informações em dados gráficos, há dois tipos básicos de objetos envolvidos nesta técnica:

- Objetos de dados (vtkDataObject);
- Objetos de processo (vtkAlgorithm).

Objetos de dados representam informações e fornecem métodos para criar, acessar, e apagar estas informações. Modificações diretas nos dados representados pelos objetos de

dados não são permitidas, exceto através de métodos de objetos formais. Esta capacidade fica reservada aos objetos de processo. Métodos adicionais também são avaliados para obter características inerentes aos dados. Isto inclui a determinação da faixa de variação dos dados, ou determinar o tamanho ou a quantidade de dados existentes no objeto.

A representação interna dos objetos de dados difere um do outro e tem significativo impacto nos métodos de acesso aos dados, bem como na eficiência do armazenamento ou performance computacional dos objetos de processo que interagem com os objetos de dados. Daí, diferentes objetos de dados podem ser usados para representar os mesmos dados dependendo da demanda por eficiência e generalidade de processo.

Os principais tipos de dados do VTK serão tratados mais adiante, por hora deve-se ter em mente que a classe vtkDataObject é uma coleção de vários tipos de dados. Dados que têm uma estrutura formal são muitas vezes referenciados como um dataset (classe vtkDataSet). Os datasets representam e permitem operações nos dados através do fluxo na cadeia de execução. O diagrama de classes da Figura 4.5 foi retirado da documentação on-line do VTK 5.0.3, nele são mostrados todos os tipos de dados derivados da classe vtkDataObject, com destaque para as seis classe que compõem a classe vtkDataSet, por serem os tipos mais comuns usados no desenvolvimento de aplicações com VTK.

Objetos de processo operam na entrada de dados para gerar dados de saída. Um objeto de processo ou deriva novos dados de sua entrada, ou transforma a entrada de dados em uma nova forma. A entrada para objetos de processo inclui um ou mais objetos de dados e/ou parâmetros locais para controlar sua operação. Parâmetros locais incluem variáveis de instância ou associações e referências para outros objetos. Por exemplo, o centro e o raio são parâmetros locais para controlar a geração da forma gráfica esférica.

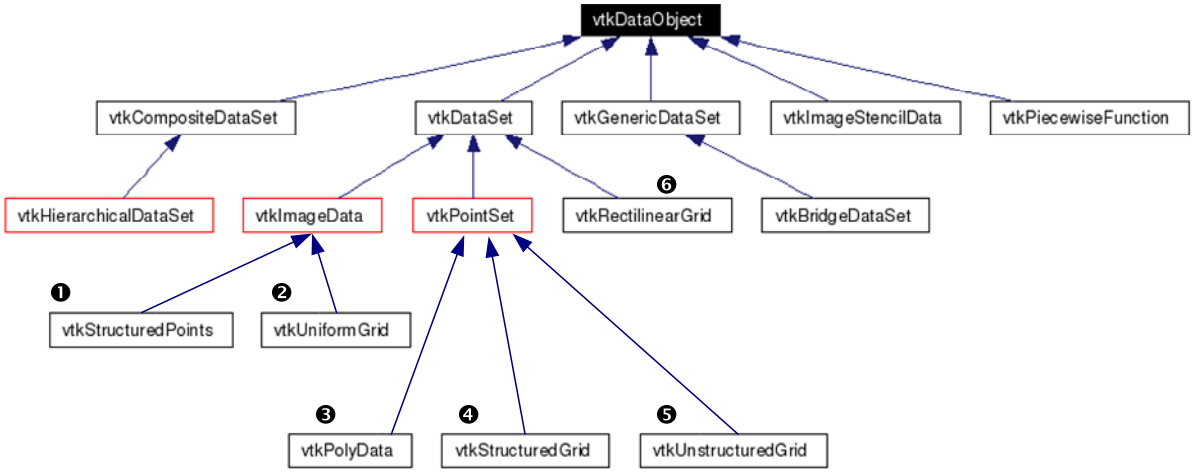


Figura 4.5 – Hierarquia da classe vtkDataObject.
Fonte: Adaptado de VTK Documentation (2007).

Objetos de processo são caracterizados como objetos gráficos (source objects), filtros (filter objects), ou objetos de mapeamento (mapper objects). Esta categorização é baseada no fato de o objeto iniciar, manter ou terminar o fluxo de dados de visualização. O próximo diagrama (Figura 4.6) ilustra o modelo de visualização do VTK, bem como os tipos de objetos, em sua forma abstrata. As flechas apontam sempre na direção do fluxo de dados.

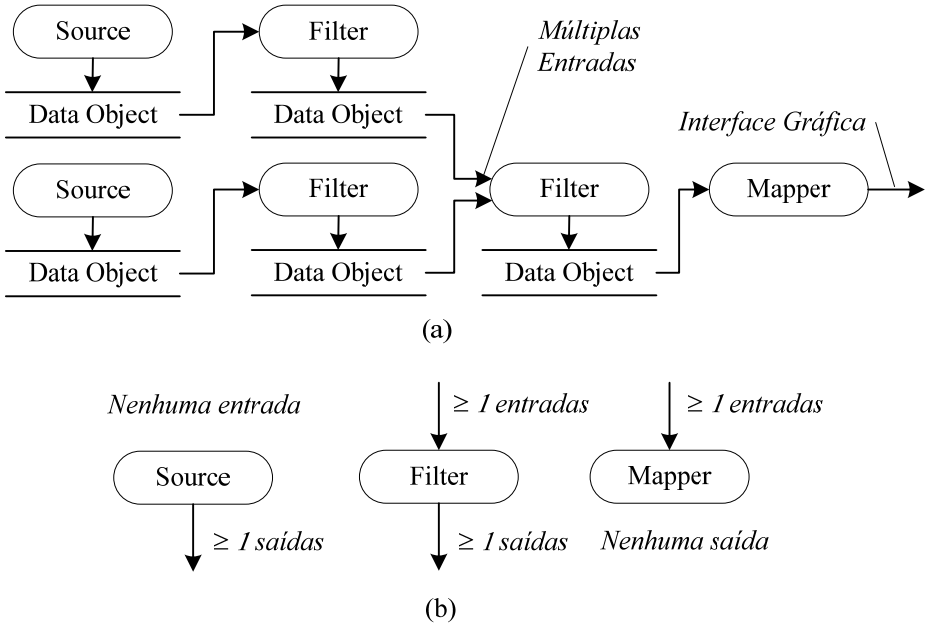


Figura 4.6 – (a) O modelo de visualização; (b) objetos de processo na sua forma abstrata.
Fonte: Adaptado de Kitware (2006).

Cabe salientar também que os objetos de processo, derivados da classe vtkAlgorithm são muitas vezes tratados simplesmente por filtros. Na versão em que este projeto foi desenvolvido, vinte e três classes derivavam diretamente de vtkAlgorithm, e a partir dessas, muitas outras. Maiores detalhes podem ser encontrados na própria documentação do VTK, em VTK Documentation (2007).

4.4.3 O Modelo de Execução

A execução do pipeline (ou network) de um modo geral pode ser resumida através do fluxograma básico da Figura 4.7.

Conforme explicado na seção anterior, o controle da execução pode ocorrer por demanda de dados ou em resposta a eventos. Pois bem, quando uma solicitação ocorre, seja qual for a ação que a tenha disparado, o renderizador inicia a requisição dos dados, então os

dados são gerados pelo source, em seguida filtrados se for o caso, mapeados e transformados em figuras, sendo representada pelo ator. Em caso de atualizações no ator, uma nova solicitação é feita e o processo se reinicia.

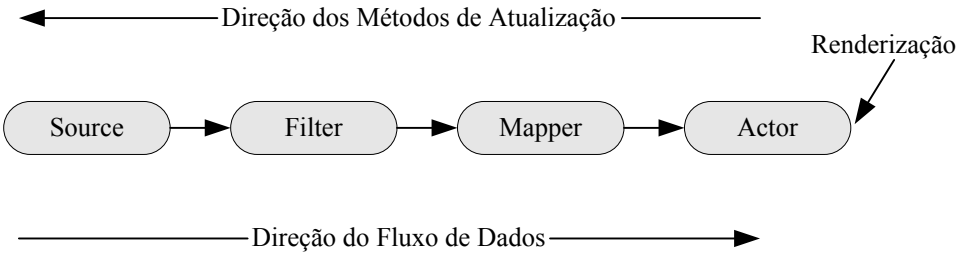


Figura 4.7 – Execução do pipeline.
Fonte: Kitware (2006).

4.5 REPRESENTAÇÃO DE DADOS

Em matemática é costume classificar as variáveis de acordo com certas características importantes. Claras distinções são feitas entre variáveis inteiras, reais e complexas ou entre variáveis representando valores escalares ou vetoriais. Esta idéia de classificação é igualmente importante em se tratando de processamento de dados, uma vez que a escolha correta da forma de representação está diretamente ligada ao desempenho da aplicação. Com este intuito, entender e classificar as estruturas de dados, as próximas seções abordam as principais formas de representar dados no VTK.

4.5.1 O Objeto de Dados

O objeto de dados do VTK é representado pela classe vtkDataObject (Figura 4.5). Um objeto de dados pode ser pensado como uma coleção de dados sem qualquer forma. Os objetos de dados representam os dados que são processados pelo pipeline de visualização, conforme já mencionado. Tomados por si só, os objetos de dados representam pouca informação útil. Isto ocorre somente quando eles passam a ser organizados dentro de algumas estruturas que fornecem os meios para poder operar com os dados, através dos algoritmos de visualização ou objetos de processo: vtkAlgorithm.

Existem vários tipos de dados que são utilizados pelo VTK, organizados sob várias estruturas. Uma dessas estruturas merece destaque por ser a mais comum e a mais utilizada na solução de problemas: o dataset ou conjunto de dados. A compreensão dessa estrutura se faz necessária para o desenvolvimento de qualquer aplicação com o VTK, além de que, a escolha da representação inadequada dos dados poderá acarretar em problemas sérios de desempenho.

4.5.2 Conjuntos de Dados

Objetos de dados com uma estrutura organizada e atributos associados constituem os conjuntos de dados (datasets) (Figura 4.8). O dataset é uma forma abstrata; sua representação e implementação foi deixada para as subclasses concretas. A maioria dos algoritmos (objetos de processo) no VTK operam com os datasets.

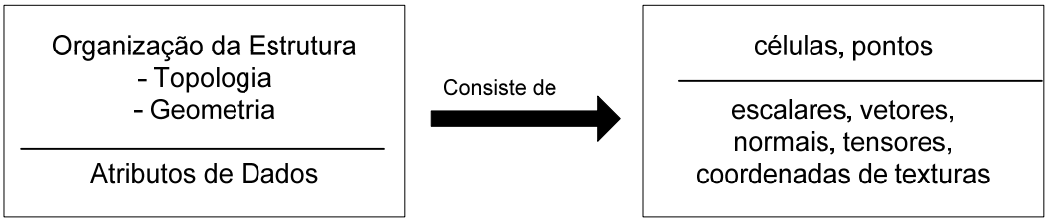


Figura 4.8 – A arquitetura de um dataset. Um dataset consiste de uma estrutura organizada, com ambas as propriedades topológicas e geométricas, e atributos de dados associados com a estrutura.
Fonte: Schroeder, Martin e Lorensen (2006).

A estrutura tem duas partes: topologia e geometria. Topologia é o conjunto de propriedades invariantes sobre certas transformações geométricas (SCHROEDER; MARTIN; LORENSEN, 2006). São consideradas transformações: rotação, translação e escalonamento não uniforme. Geometria é a instanciação da topologia, a especificação da posição no espaço tridimensional. Por exemplo, dizer que um polígono é um "triângulo" especifica a topologia. Fornecendo as coordenadas dos pontos estamos especificando a geometria.

Atributos do dataset são informações suplementares associadas com a geometria e/ou topologia. Estas informações podem ser o valor da temperatura em um ponto ou a massa inercial de uma célula.

O modelo de um dataset assume que a estrutura consiste de células e pontos. As células especificam a topologia, enquanto que os pontos especificam a geometria. Atributos típicos incluem escalares, vetores, normais, coordenadas de textura e tensores.

A definição da estrutura de um dataset como uma coleção de células e pontos é uma conseqüência direta da natureza discreta de seus dados. Em se tratando de elementos finitos é possível estabelecer uma correspondência entre pontos e nós, bem como entre células e elementos, ou seja:

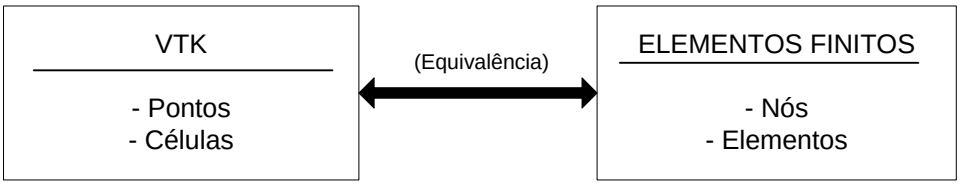


Figura 4.9 – Relação VTK x Elementos Finitos.
Fonte: O autor.

A identificação dessa equivalência foi importante para o desenvolvimento do programa *bioMeshCreate*, onde os dados visualizados deveriam ser exportados para softwares de análise por elementos finitos, como o Ansys Multiphysics (2007).

Existem seis tipos definidos de dataset no VTK: *dados poligonais*, *dados de imagem*, *malha retilínea*, *malha estruturada*, *pontos não-estruturados* e *malha não-estruturada*. Para entender como estão organizados, no entanto, é necessário ter uma visão sobre os tipos de células. As próximas seções tratam então, dos tipos de células e datasets existentes.

4.5.3 Células

Um dataset consiste de uma ou mais células. Células são os blocos fundamentais de sistemas de visualização. Sob a óptica de elementos finitos, são referenciadas como elementos. Células são definidas especificando-se um tipo em combinação com uma lista ordenada de pontos. Esta lista ordenada é comumente chamada de lista de conectividade, ou incidência nodal no caso de elementos finitos. Combinada com a especificação de tipo, implicitamente define a topologia da célula. As coordenadas x-y-z dos pontos definem a geometria da célula.

Para facilitar a compreensão, considere a célula de exemplo, mostrada na Figura 4.10. A figura mostra uma célula do tipo hexaedro. A lista ordenada é uma seqüência de ids (identificadores) de pontos indexados em relação à lista de coordenadas dos pontos. A topologia desta célula é implicitamente conhecida: sabe-se, por exemplo, que (8,10) é uma das 12 arestas do hexaedro, e que (8,10,22,21) é uma de suas seis faces.

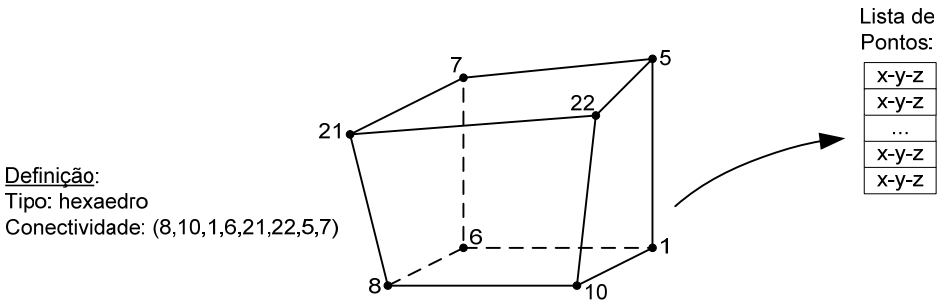


Figura 4.10 – Exemplo de célula hexaédrica. A topologia é implicitamente definida pela lista ordenada de pontos
Fonte: Schroeder, Martin e Lorensen (2006).

4.5.4 Tipos de Célula

No VTK as células classificam-se em lineares e não-lineares.

Lineares. As células lineares são aquelas que possuem função de interpolação constante ou linear. A Figura 4.11 ilustra os dezesseis tipos de células lineares atualmente presentes no VTK. Algumas delas, por serem mais frequentemente usadas, serão comentadas. Importante notar que a maioria delas são primárias, enquanto que outras são formadas pela composição de células primárias.

O vértice é uma célula primária adimensional, definida por um simples ponto.

O polivértice é uma composição de células adimensionais definidas por uma lista arbitrária ordenada de pontos.

A linha é uma célula primária unidimensional definida por dois pontos. A direção ao longo da linha é dada do primeiro para o segundo ponto.

Polilinha é uma composição de células unidimensionais consistindo de uma ou mais linhas conectadas.

Triângulo é a mais simples das células bidimensionais.

A faixa de triângulos (triangle strip) é uma composição de células bidimensionais, consistindo de um ou mais triângulos.

O quadrilátero é uma célula primária bidimensional, definida por uma lista ordenada de quatro pontos formando um plano.

O pixel é uma célula primária bidimensional definida por uma lista ordenada de quatro pontos, sendo topologicamente equivalente ao quadrilátero, porém com uma restrição

geométrica: cada aresta do pixel é perpendicular à sua aresta adjacente e paralela a um dos eixos coordenados x-y-z, daí a normal para o pixel é também paralela a um dos eixos coordenados.

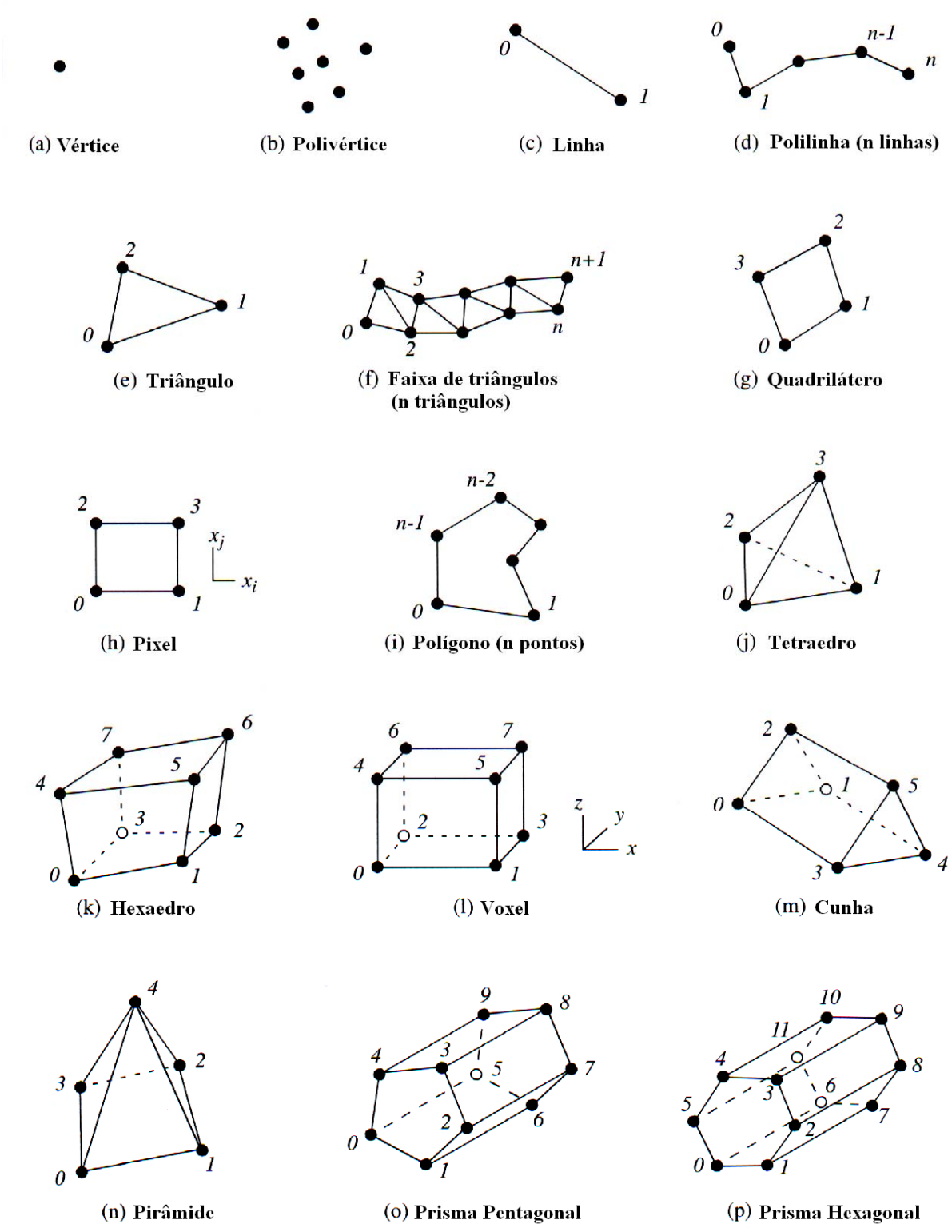


Figura 4.11 – Células lineares encontradas no VTK.
Fonte: Schroeder, Martin e Lorensen (2006).

O polígono é uma célula primária bidimensional, definido por uma lista ordenada de três ou mais pontos dispostos em um plano. O polígono pode ser não-convexo, mas não pode ter regiões internas fechadas e nem se auto-intersectar.

O tetraedro é uma célula primária tridimensional, definida por uma lista de quatro pontos não-planares, possui seis arestas e quatro faces triangulares.

O hexaedro é uma célula primária tridimensional consistindo de seis faces quadriláteras, doze arestas e oito vértices. Deve ser convexo.

Voxel é uma célula primária tridimensional, topologicamente equivalente ao hexaedro com uma restrição geométrica adicional: cada face do voxel é perpendicular a um dos eixos coordenados x-y-z. Ele é um caso particular do hexaedro e é usado para fornecer maior desempenho computacional.

Além destas existem ainda as células lineares em forma de cunha, pirâmide, prisma pentagonal e prisma hexagonal.

Não-lineares. As células não-lineares, isto é, células que usam funções não-lineares para sua formulação, normalmente por combinações de polinômios, são bastante comuns em análises numéricas, pois são mais precisas e se adaptam com maior facilidade à geometrias curvas, no entanto, o número de possíveis funções-base não-lineares é ilimitado, o que ocasiona um sério problema combinatorial para qualquer sistema de visualização, ou seja, não é possível implementar todos os tipos de células não-lineares. Para solucionar este problema, o VTK possui duas abordagens. A primeira é dar suporte diretamente aos tipos de células não-lineares com funções de interpolação quadrática. Tais células são construídas adicionando-se nós nos pontos médios das arestas e eventualmente no interior e centro das faces, necessitando assim estender a lista de conectividade para refletir a adição das entradas extras. A segunda é através do seu framework para adaptação de células, que permite ao usuário interfacear qualquer função-base através de um sistema paramétrico de coordenadas. Mais detalhes sobre esta abordagem pode ser encontrado em Schroeder, Martin e Lorensen (2006).

As células não-lineares atualmente disponíveis no VTK, interpoladas por funções quadráticas, são em número de treze, todas primárias, e podem ser observadas na Figura 4.12.

Uma diferença significante entre células lineares e não-lineares é o modo como elas são renderizadas e operadas por vários algoritmos de visualização. Células lineares são prontamente convertidas para primitivas gráficas lineares, as quais são então processadas pela biblioteca gráfica. Células não-lineares, por outro lado, não têm suporte direto na biblioteca gráfica (uma exceção são as da família de *não-uniforme racional B-splines* ou NURBS).

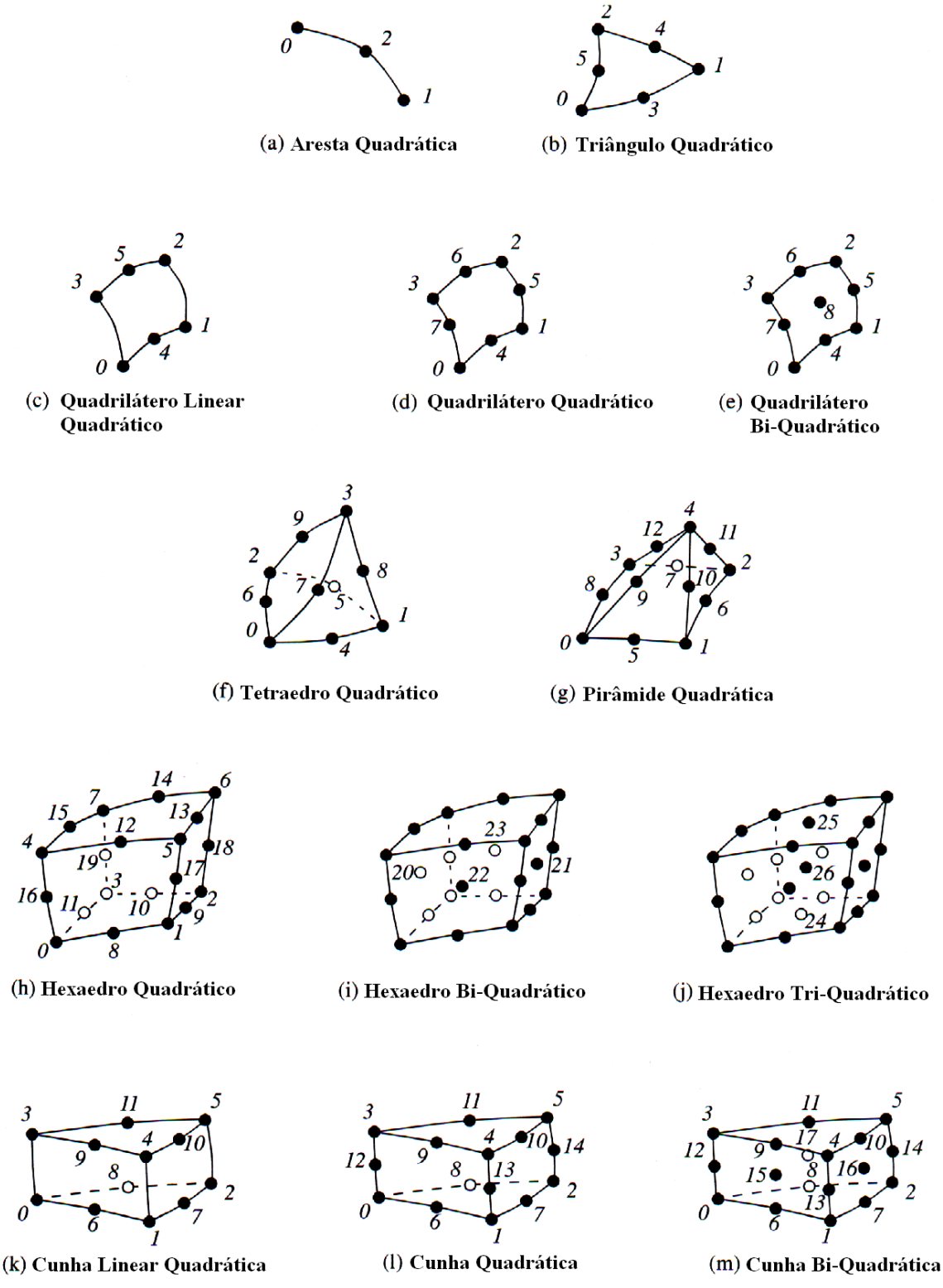


Figura 4.12 – Células não-lineares encontradas no VTK.
Fonte: Schroeder, Martin e Lorensen (2006).

Para serem processadas, as células não-lineares devem ser especialmente tratadas pelo sistema de visualização e isto inclui: 1) Divisão das células não-lineares em células lineares, passando a operar de forma linear. 2) Desenvolvimento de algoritmos de

visualização e renderização para operar diretamente em células não-lineares. 3) Customização das operações de renderização diretamente na biblioteca gráfica. Essas questões são tópicos atuais na área de visualização (SCHROEDER; MARTIN; LORENSEN, 2006).

4.5.5 Tipos de Conjunto de Dados

Conforme já definido, um conjunto de dados (dataset) consiste de uma estrutura organizada mais atributos de dados associados. A estrutura possui propriedades topológicas e geométricas e é composta de um ou mais pontos e células. O tipo de um dataset deriva da organização de sua estrutura, e especifica o relacionamento que as células e pontos têm entre si. Os tipos de dataset são mostrados na Figura 4.13 e serão discutidos na seqüência, uma vez que sua utilização adequada está diretamente ligada ao desempenho da aplicação.

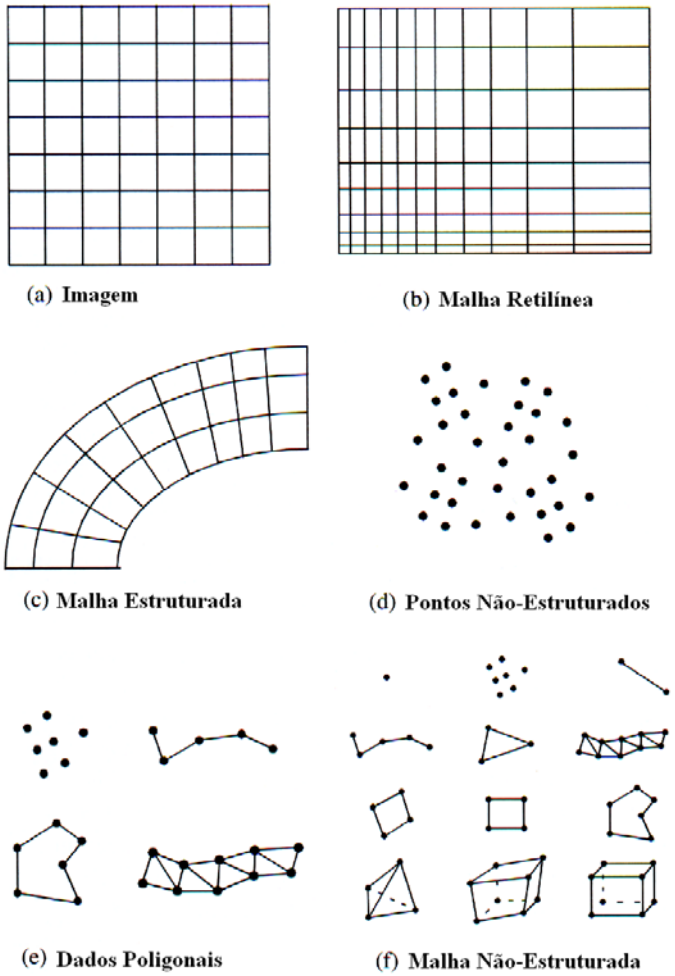


Figura 4.13 – Tipos de dataset. A malha não-estruturada consiste de todos os tipos de células.
Fonte: Schroeder, Martin e Lorensen (2006).

4.5.5.1 Dados de imagem

Um dataset do tipo imagem é uma coleção de pontos e células arranjadas regularmente em uma grade retangular. As linhas, colunas, e planos da grade são paralelos ao sistema global de coordenadas x-y-z. Se os pontos e células são arranjados em um plano (bidimensional) o dataset é referenciado como um mapa de pixels (pixmap), mapa de bits (bitmap) ou imagem. Se os pontos e células são arranjados como planos empilhados (tridimensional) o dataset é referenciado como um volume.

Dados de imagem consistem de elementos de linha (1D), pixels (2D), ou voxels (3D). É regular em ambos: geometria e topologia, e podem ser implicitamente representados. O esquema de representação requer somente dimensões dos dados, um ponto de origem e o espaçamento dos dados.

4.5.5.2 Malha retilínea

A malha retilínea é uma coleção de pontos e células arranjadas em uma grade regular. As linhas, colunas e planos da grade são paralelos ao sistema global de coordenadas x-y-z. Enquanto a topologia é regular, a geometria é somente parcialmente regular, isto é, os pontos estão alinhados ao longo do eixo de coordenadas, mas o espaço entre os pontos pode variar.

Assim como o dataset do tipo imagem, as malhas retilíneas consistem de pixels (2D) ou voxels (3D). A topologia é representada implicitamente pela especificação da dimensão da malha. A geometria é representada mantendo-se uma lista separada das coordenadas x, y e z. Para obter as coordenadas de um ponto em particular, os valores de cada uma das três listas devem ser apropriadamente combinados.

4.5.5.3 Malha estruturada

A malha estruturada é um dataset com topologia regular e geometria irregular. A malha pode ser curvada em qualquer configuração nas quais as células não se sobrepõem ou se auto-intersectam. A topologia da malha estruturada é representada implicitamente por um

vetor de três dimensões (n_x, n_y, n_z). A geometria é explicitamente representada mantendo-se um vetor de coordenadas dos pontos. As células que constituem uma malha estruturada são quadriláteros (2D) ou hexaedros (3D).

Malhas estruturadas são comumente encontradas em análise de diferenças finitas. Diferença finita é uma técnica de análise numérica para aproximar a solução por equações diferenciais parciais. Aplicações típicas incluem fluxo de fluidos, transferência de calor e combustão.

4.5.5.4 Pontos não-estruturados

Pontos não-estruturados são pontos irregularmente distribuídos no espaço. Não há topologia em um dataset de pontos não-estruturados, e a geometria é completamente não-estruturada. Os vértices e polivértices são usados para representá-los.

Pontos não-estruturados são simples, porém um importante tipo de dataset. Servem para representar dados não-estruturados. Tipicamente, esta forma é transformada em outra mais estruturada para propósito de visualização. Existem algoritmos responsáveis por esta transformação no VTK.

4.5.5.5 Dados poligonais

O dataset poligonal consiste de vértices, polivértices, linhas, polilinhas, polígonos e faixas de triângulos. A topologia e geometria dos dados poligonais é não-estruturada, e as células que compõem este dataset variam em dimensão topológica. O dataset poligonal forma uma ponte entre dados, algoritmos e computação gráfica de alta velocidade.

Vértices, linhas e polígonos formam um conjunto mínimo de primitivas para representar 0-, 1- e 2-dimensões geométricas. Polivértices, polilinhas e faixa de triângulos são incluídos por questão de compactação e velocidade. Faixas de triângulos em particular são primitivas de alta velocidade. Para representar n triângulos com uma faixa de triângulos são necessários n+2 pontos, comparado aos 3n pontos para a representação convencional. Além do mais, muitas bibliotecas gráficas podem renderizar faixas de triângulos em velocidade maior do que polígonos triangulares.

4.5.5.6 Malha não-estruturada.

A forma mais geral de um dataset é a malha não-estruturada. Tanto a topologia quanto a geometria são completamente não-estruturadas. Qualquer tipo de célula pode ser arbitrariamente combinada neste tipo de dataset. Daí a topologia das células variam de 0D (vértices, polivértices) a 3D (tetraedros, hexaedros, voxels). No VTK qualquer dataset pode ser expresso em termos de malha não-estruturada, mas deve ser usado somente quando for estritamente necessário, pois este é o tipo de dataset que requer a maior quantidade de memória e recursos computacionais para representar e operar com os dados.

Malhas não-estruturadas são encontradas em campos como a análise por elementos finitos, geometria computacional e modelagem geométrica. Análise por elementos finitos é uma técnica de solução numérica pelo uso de equações diferenciais parciais (EDPs). Aplicações deste tipo de análise incluem cálculo estrutural (estruturas biomecânicas, por exemplo), vibrações, dinâmica, transferência de calor, etc. Comparada à análise de diferenças finitas, a vantagem da análise por elementos finitos é que a restrição em relação à topologia regular é removida, daí domínios complexos podem ser mais facilmente gerados (malha). A Figura 4.14 ilustra essa situação, onde uma malha não-estruturada foi gerada a partir de um conjunto aleatório de pontos no plano, através da triangulação de Delaunay 2D. A imagem foi gerada com o programa bioMeshCreate.

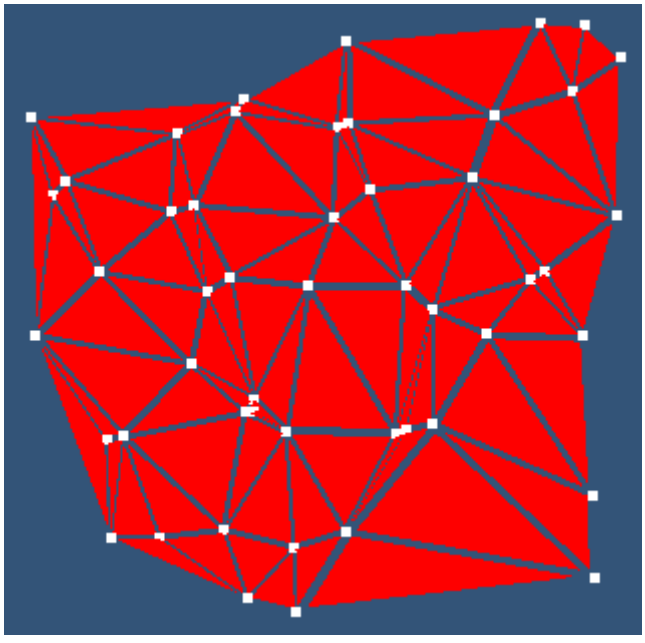


Figura 4.14 – Exemplo de uma malha não-estruturada, gerada com bioMeshCreate.
Fonte: O autor.

4.6 MANIPULAÇÃO DE ARQUIVOS

O Visualization Toolkit fornece várias classes para manipular arquivos que permitem ler e escrever em arquivos nos mais populares formatos. Dentre estes formatos encontram-se: BYU, PDB, PLY, STL, XML, BMP, DICOM, JPEG, PNM, PNG, SLC, TIFF, PLOT3D, POP, AVS, CGM. E também aqueles que guardam informações de toda a cena, incluindo geometria, câmera, luzes, atores e propriedades, tais como 3D Studio, VRML, PostScript, Inventor, Wavefront e GeomView.

Além desses, o VTK possui também o seu próprio formato. A razão principal em criar um formato próprio é fornecer um esquema consistente de representação de dados para uma variedade de tipos de conjuntos de dados, e permitir a comunicação de dados entre outros sistemas. Existem dois tipos de arquivos avaliados no VTK. O formato mais simples é o legacy, formato serial que é fácil de ler e escrever tanto manual como programaticamente. Todavia, o formato legacy é menos flexível que o outro formato: o XML. O formato XML suporta acesso randômico, E/S paralela, e compressão portável de dados. Neste projeto optou-se pelo formato XML.

Na verdade, o programa bioMeshCreate disponibiliza os formatos BMP, JPG e PNG para dados de imagem. Para conjuntos de dados, o VTK XML. Para malha de triângulos, o formato STL. Para exportação da cena, o formato de realidade virtual: VRML. E para as imagens médicas utiliza o formato DICOM e RAW-16. As próximas seções abordam resumidamente os formatos implementados.

4.6.1 Formatos de Imagem

Dentre os vários formatos de imagem existentes, foram implementados no bioMeshCreate, os formatos BMP, JPG e PNG. Como esses formatos são bastante conhecidos no universo da computação, somente um breve comentário será feito.

O formato BMP ou mapa de bits é um dos formatos mais conhecidos para armazenamento de imagens. Apresenta uma excelente resolução, porém seu uso é desencorajado na transmissão de dados por computador, devido ao tamanho dos arquivos

gerados, normalmente muito grandes.

O formato JPG ou JPEG foi criado pelo Joint Photographic Experts Group, tratando-se de um formato de compressão, com perda de dados, aplicado em imagens fotográficas. A perda de dados é proporcional ao fator de compressão desejado.

PNG (Portable Network Graphics) é um formato de dados utilizado para imagens, que surgiu em 1996, como substituto para o formato GIF, devido ao fato de este último incluir algoritmos patenteados. Esse formato livre é recomendado pela W3C (World Wide Web Consortium), suporta canal alfa, tem maior gama de profundidade de cores, alta compressão (regulável), além de outras características. O formato PNG permite comprimir as imagens sem perda de qualidade e retirar o fundo branco de imagens, ao contrário do que acontece com outros formatos, como o JPG, por isso é um formato válido para imagens que precisam manter 100% da qualidade para reuso (WIKIPEDIA, 2007).

4.6.2 O Formato XML

O XML (eXtensible Markup Language – linguagem de marcação extensível), foi criado em 1996 pelo W3C para ser uma metalinguagem flexível, mas formal, para ser usado na Internet. Uma metalinguagem é uma linguagem que serve para descrever outras (BROGDEN; MINNICK, 2002). Esta habilidade permite que os aplicativos acessem dados com base em tags (marcas) descritivas ao invés de apenas pela posição dos dados. Além disso, pelo fato de ser extensível, permite a inserção de tags personalizadas dentro do documento.

A listagem seguinte (Figura 4.15), é um exemplo de arquivo XML gerado com bioMeshCreate, representando a geometria de um cubo. A tag estendida: bioMeshCreate, logo no início do documento representa o estado do objeto (cubo) no momento em que ele foi salvo. Por questão de clareza na apresentação, alguns nós e atributos foram omitidos.

Cada tag presente no arquivo representa uma informação que segue um esquema de dados. Informações detalhadas sobre este esquema podem ser encontradas em Kitware (2006). Além disso, todos os arquivos XML do VTK são documentos bem formados.

```
<?xml version='1.0'?>
<VTKFile version="0.1" byte_order="LittleEndian" type="PolyData" >
  <bioMeshCreate ObjectType="4" ObjectName="Cube_1" >
    <cbbRepresentation>2</cbbRepresentation>
    <chbVisible>1</chbVisible>
    ...
  </bioMeshCreate>
  <PolyData>
    <Piece NumberOfStrips="0" NumberOfPoints="24" NumberOfLines="0" >
      <PointData Normals="Normals" TCoords="TCoords" >
        <DataArray NumberOfComponents="3" format="ascii" type="Float32">
          -1 0 0 -1 0 0
          -1 0 0 -1 0 0
          ...
        </DataArray>
        <DataArray NumberOfComponents="2" format="ascii" type="Float32">
          0 0 1 0 0 1
          1 1 -0 0 -1 0
          ...
        </DataArray>
      </PointData>
      <CellData/>
      <Points>
        <DataArray NumberOfComponents="3" format="ascii" type="Float32">
          -0.5 -0.5 -0.5 -0.5 -0.5 0.5
          -0.5 0.5 -0.5 -0.5 0.5 0.5
          ...
        </DataArray>
      </Points>
      <Verts>
        <DataArray format="ascii" type="Int32" Name="connectivity" />
        <DataArray format="ascii" type="Int32" Name="offsets" />
      </Verts>
      <Lines>
        <DataArray format="ascii" type="Int32" Name="connectivity" />
        <DataArray format="ascii" type="Int32" Name="offsets" />
      </Lines>
      <Strips>
        <DataArray format="ascii" type="Int32" Name="connectivity" />
        <DataArray format="ascii" type="Int32" Name="offsets" />
      </Strips>
      <Polys>
        <DataArray format="ascii" type="Int32" Name="connectivity" >
          0 1 3 2 4 6
          7 5 8 10 11 9
          ...
        </DataArray>
        <DataArray format="ascii" type="Int32" Name="offsets" >
          4 8 12 16 20 24
        </DataArray>
      </Polys>
    </Piece>
  </PolyData>
</VTKFile>
```

Figura 4.15 – Listagem de um arquivo XML contendo a geometria de um cubo, gerado com bioMeshCreate.
Fonte: O autor.

Segundo Wahlin (2003) um documento XML é bem formado quando segue os seguintes princípios básicos:

- Contém somente um elemento raiz, neste caso, o elemento VTKFile.
- Todos os elementos devem ter uma tag de fechamento.
- Todos os valores de atributos devem estar entre aspas.
- Embora os elementos XML possam conter elementos filhos, todos os elementos precisam estar devidamente aninhados.
- Diferenças entre maiúsculas e minúsculas devem ser consideradas nos nomes de elementos.

Uma vez garantido que o documento XML é bem formado, sua manipulação pode ser realizada de maneira programática através dos analisadores sintático-morfológicos (parsers) presentes em grande parte das linguagens de programação hoje em dia, facilitando assim a pesquisa no documento, inserção de novas tags e atributos. O programa bioMeshCreate utilizou o analisador de documentos XML presente no Qt 4.3.0 da TrollTech[3], segundo a arquitetura DOM (Document Object Model). O XML DOM é uma especificação da W3C e serve como interface de acesso para os diversos elementos que compõem um documento XML.

4.6.3 O Formato STL

O Formato STL (STereoLitography) é o formato mais utilizado como interface entre os processos de prototipagem rápida (RP). Vários formatos de imagem (arquivos) interpretados pelos processos de RP podem ser citados, mas o formato STL é aceito como padrão, por ser um formato aberto e simples. Souza, Centeno e Pedrini (2003) explicaram a estrutura de dados contida num arquivo STL. O próximo parágrafo resume sua explanação.

No formato STL, a superfície é subdividida em vários mosaicos formando uma série de pequenos triângulos (faces) (BURNS, 1993; KUMAR; DUTTA, 1997). O arquivo STL consiste de uma lista com os dados das faces triangulares, no qual cada face é unicamente

[3] O Qt foi a ferramenta utilizada para desenvolver a interface gráfica do programa bioMeshCreate. Trata-se de um framework multi-plataforma, de código-livre, em C++.

identificada por uma normal unitária (um vetor perpendicular ao triângulo cujo comprimento é uma unidade arbitrária) e por três pontos representando os vértices do triângulo. Tanto a normal como cada vértice são especificados por três coordenadas (x, y, z), portanto, tem-se 12 valores para cada face. As faces definem a superfície de um objeto tridimensional. Cada face faz parte do limite entre o interior e o exterior do objeto. A orientação das faces é especificada de duas formas que devem ser consistentes: 1) a direção da normal de cada triângulo é para fora (Figura 4.16), tal que a orientação das faces no modelo proposto estejam todas voltadas para o lado de fora da peça, ou seja, o lado que não possui material na peça; 2) os vértices são listados na direção anti-horária (quando o objeto é visualizado a partir do lado de fora); neste caso aplica-se a regra da mão direita (Figura 4.16) (BURNS, 1993).

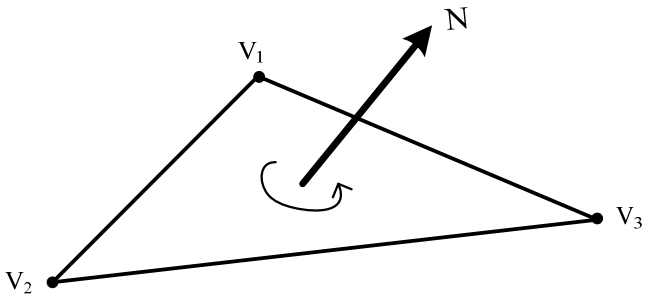


Figura 4.16 – Representação de dados no formato STL.
Fonte: Souza, Centeno e Pedrini (2003).

4.6.4 Formato de Realidade Virtual

VRML (2008) – *Virtual Reality Modeling Language*, ou Linguagem para Modelagem de Realidade Virtual, é um padrão de aplicativos de realidade virtual utilizado na Internet. Por meio desta linguagem, escrita em modo texto, é possível criar objetos tridimensionais podendo definir cor, transparência, brilho e textura. Usualmente a extensão para esta linguagem é WRL. A VRML 2.0 permite adicionar áudio, vídeo, e integração com linguagens de script. É, essencialmente, um arquivo de dados, mas com a capacidade de simular realidade e imersão em ambientes. O VTK possibilita exportar a cena para arquivos no formato VRML 2.0.

4.6.5 Formatos de Imagem Médica

O formato DICOM, abreviação de *Digital Imaging Communications in Medicine* (ou comunicação de imagens digitais em medicina), é um conjunto de normas para tratamento, armazenamento e transmissão de informação médica (imagens médicas) num formato eletrônico, que tipicamente contém um cabeçalho onde estão armazenados os dados da imagem como tamanho, resolução, informações do paciente, equipamento em que foi feita, taxa de compressão, se houver, entre outras. O problema é que o tamanho desse cabeçalho varia de arquivo para arquivo, de acordo com o equipamento que o gerou. Além disso, como a taxa de compressão pode variar, o número de bits por pixel nesse tipo de arquivo também é variável. Uma outra complicação é que algumas vezes, um ou mais bits em cada pixel é usado para marcar a conectividade entre voxels. Devido a essas diferenças, fica difícil e até inviável programar um algoritmo generalizado de reconstrução de imagens que utilize os arquivos DICOM como entrada. A solução é, portanto, converter previamente para um outro formato, conhecido como RAW.

RAW é um formato de arquivo digital de imagens que contém a totalidade dos dados da imagem tal como foi originalmente captada, sem processamento adicional. É muito utilizado pelos fabricantes de câmeras fotográficas digitais. O formato RAW não costuma levar aplicada a compressão como ocorre com o popular JPEG, nem contém qualquer cabeçalho de dados. No VTK, o padrão adotado é de 16-bits/pixel e a classe responsável pela leitura desse tipo de arquivo e posterior geração do volume é a vtkVolume16Reader.

4.7 UTILIZAÇÃO DO VTK

Baseado na experiência adquirida durante a fase de pesquisa e implementação do programa bioMeshCreate com o uso do VTK, será apresentado a seguir, algumas de suas principais vantagens e desvantagens.

4.7.1 Vantagens

- É uma ferramenta de código-livre.
- É multi-plataforma.
- Conta com uma experiência de quase quinze anos na arte de visualização científica, tendo iniciado suas atividades em dezembro de 1993.
- Oferece um bom material para estudo, entre livros, artigos e fóruns oficiais de discussão.
- Face a outros sistemas de visualização como o Matlab, com sua toolbox de processamento de imagens, oferece a vantagem de gerar aplicações que podem ser executadas de maneira independente (stand-alone) e livremente distribuídas.

4.7.2 Desvantagens

- Requer um bom equipamento para sua execução, em contrapartida ao grande número de recursos que oferece. Embora isso seja uma desvantagem, hoje em dia, com o rápido avanço da tecnologia e hardwares cada vez mais sofisticados, não tem sido um grande empecilho.
- Para o desenvolvimento de aplicações mais avançadas, é recomendado um conhecimento mínimo de orientação a objeto, uma vez que todas as classes do VTK encontram-se organizadas segundo este paradigma.
- Tem um tempo de desenvolvimento considerável, principalmente em virtude da curva de aprendizagem, pois o número de recursos que disponibiliza é imenso, porém compreendê-los e combiná-los de forma adequada nem sempre é tarefa fácil.

Após vislumbrar o potencial dessa ferramenta, analisando os prós e os contras, entendemos ser viável a sua utilização como a biblioteca gráfica-base para implementação do programa computacional bioMeshCreate, objeto de estudo do próximo capítulo.

5 O PROGRAMA COMPUTACIONAL BIOMESHCREATE

O presente capítulo tem por objetivo apresentar o programa bioMeshCreate, que foi desenvolvido como parte integrante deste trabalho. O apêndice A, no final do trabalho, traz os pré-requisitos e os passos necessários para sua instalação sob a plataforma Windows, a partir do CD-ROM.

Também no CD-ROM encontra-se disponível o manual do usuário do programa, em formato pdf. O manual abrange todos os recursos existentes no aplicativo, seu funcionamento, possíveis configurações, ferramentas auxiliares, formas gráficas, filtros, módulo de elementos finitos e persistência de dados. Portanto, neste capítulo serão apresentados basicamente os recursos relacionados à reconstrução de imagem, geração de malha e integração com softwares de análise por elementos finitos. Para maiores detalhes o manual do usuário poderá ser consultado.

5.1 O QUE É BIOMESHCREATE

O bioMeshCreate (ou simplesmente bioMesh) é um programa computacional multi-plataforma, orientado a objetos, desenvolvido em C++ com o uso do VTK, e seu principal objetivo é reconstruir volumes a partir de imagens planas, normalmente imagens médicas no formato DICOM. Gerar malhas de superfície nos sólidos reconstruídos, e em alguns casos também a malha de volume, permitindo posterior exportação para formatos compatíveis com softwares de análise por elementos finitos, como o Ansys Multiphysics. Além de gerar a malha, o bioMesh permite a aplicação de vários filtros sobre as imagens geradas, tais como a simplificação da malha de triângulos, suavização de superfícies, extração de arestas, redução poligonal, entre outros (JÓIA FILHO, 2008).

O programa ainda conta com duas ferramentas auxiliares desenvolvidas. A primeira para conversão e visualização das fatias bidimensionais, a ferramenta *DicomConverter*. A segunda para integrar os modelos gerados diretamente no Ansys, a ferramenta *AnsysControl*. Estas ferramentas serão detalhadas mais adiante, neste capítulo.

Inicialmente o bioMeshCreate está disponível para a plataforma Microsoft Windows 32-bits, podendo vir a ser recompilado e disponibilizado para outras plataformas no futuro.

5.2 ARQUITETURA DO PROGRAMA BIOMESHCREATE

O bioMeshCreate consiste, quanto à arquitetura, de três camadas (ver Figura 5.1), sendo que as duas primeiras representam a contribuição do VTK.

A primeira camada (núcleo) contém código C++ *standard*, presente em praticamente todos os compiladores C. Este padrão, conhecido como C++ ANSI (American National Standards Institute) pode ser encontrado em C++ Forum (1995). Essa é uma das estratégias usadas pelo VTK para alcançar a portabilidade do código, conforme já explicado no capítulo anterior (seção 4.2).

A segunda camada deve ser convertida com o auxílio do programa CMake para a linguagem e o compilador escolhidos, dentre os disponíveis. Esta ação contribui para manter a portabilidade entre plataformas e compiladores. Maiores detalhes em Martin e Hoffman (2006). Neste projeto, o compilador VC8 da Microsoft foi escolhido por ser um dos mais usados. Além disso, o Visual C++ 2005 Express Edition (VC8) é disponibilizado pela Microsoft para ser usado gratuitamente pelo período de um ano. Em Microsoft (2007) podem ser encontrados os links para download e também instruções para instalação.

A terceira camada, também em C++, é a responsável pela interface gráfica do usuário (GUI), e foi desenvolvida com o auxílio da ferramenta Qt 4.3.0 da Trolltech. Trata-se de um framework multi-plataforma (Windows, Linux, Unix, Mac), de código fonte aberto, livremente disponível em Qt (2007).

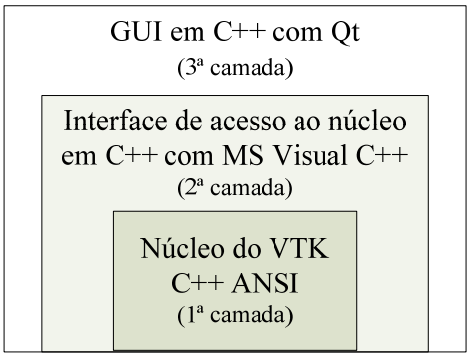


Figura 5.1 – Esquema da arquitetura do programa bioMeshCreate.
Fonte: O autor.

Conforme descrito, as ferramentas utilizadas foram obtidas sem a necessidade de licença de uso, completando um ambiente de desenvolvimento livre. Além disso, todas as ferramentas utilizadas, exceto o MS Visual C++ 2005 (mas que pode ser substituído graças ao

conversor CMake) são ferramentas multi-plataformas. Desse modo, torna-se efetivamente possível a compilação do programa em outra plataforma, como por exemplo, o Linux. Neste caso, todas as ferramentas devem ser baixadas, convertidas e recompiladas usando o ambiente da plataforma específica. No caso do compilador C deve-se utilizar outro, como por exemplo, o GCC (2007).

5.3 ORGANIZAÇÃO DO PROGRAMA

O programa bioMeshCreate pode ser dividido, quanto à organização, em: 1) sistema de menus; 2) barras de ferramentas; 3) pipeline browser; 4) object inspector; 5) janela de renderização e 6) barra de status (Figura 5.2). Os próximos parágrafos elucidam de forma resumida esta organização.

O *sistema de menus* é constituído por nove grupos. Esses grupos estão organizados segundo a sua funcionalidade, assim como os itens contidos em cada um deles. O grupo 'Formas Gráficas' permite criar formas geométricas tridimensionais simples, como o cilindro, cone, esfera ou cubo, conjuntos de pontos aleatórios em duas e três dimensões, e formas 3D reconstruídas a partir de imagens planas. O grupo 'Filtros' admite as seguintes operações: triangulação de Delaunay 2D e 3D, geração de malha triangular, simplificação de malha triangular, suavização de superfícies, extração de arestas, politubos, glifos, redução poligonal e obtenção do contorno da imagem.

As *barras de ferramentas* contêm as principais funcionalidades presentes nos menus, agrupadas, com a finalidade exclusiva de fornecer um atalho para a execução das tarefas consideradas mais comuns. Na versão atual existem seis barras de ferramentas e sua exibição pode ser controlada pelo menu exibir.

O *pipeline browser* é uma janela flutuante que contém a lista de todos os objetos renderizados pelo programa até aquele momento. Essa lista tem a forma de uma árvore (tree view), onde os objetos aparecem organizados de maneira hierárquica. É através desse objeto que as operações de edição e/ou exclusão de elementos podem ser realizadas, como por exemplo, aumentar o raio de um cilindro ou excluir algum filtro.

O *object inspector* é uma janela flutuante responsável por exibir todos os atributos de um elemento gráfico, através dos componentes da GUI (Graphic User Interface), tais como caixas de texto, caixas combinadas, caixas de checagem, barras de rolagem, entre outros. Por

ser extensa a lista de atributos de um objeto, esta janela foi dividida em três guias durante a fase de projeto: guia propriedades; guia exibição e guia informações.

A *janela de renderização* é fixa. Nela os objetos são renderizados. Normalmente se apresenta na cor azul, podendo ser modificada pelo usuário através das opções do sistema, localizada no menu ferramentas.

A *barra de status*, última região a ser apresentada, serve para prestar breves esclarecimentos sobre uma determinada função ou seu andamento. Do lado direito costuma mostrar a última posição, em pixels, ocupada pelo cursor, em relação à janela de renderização.

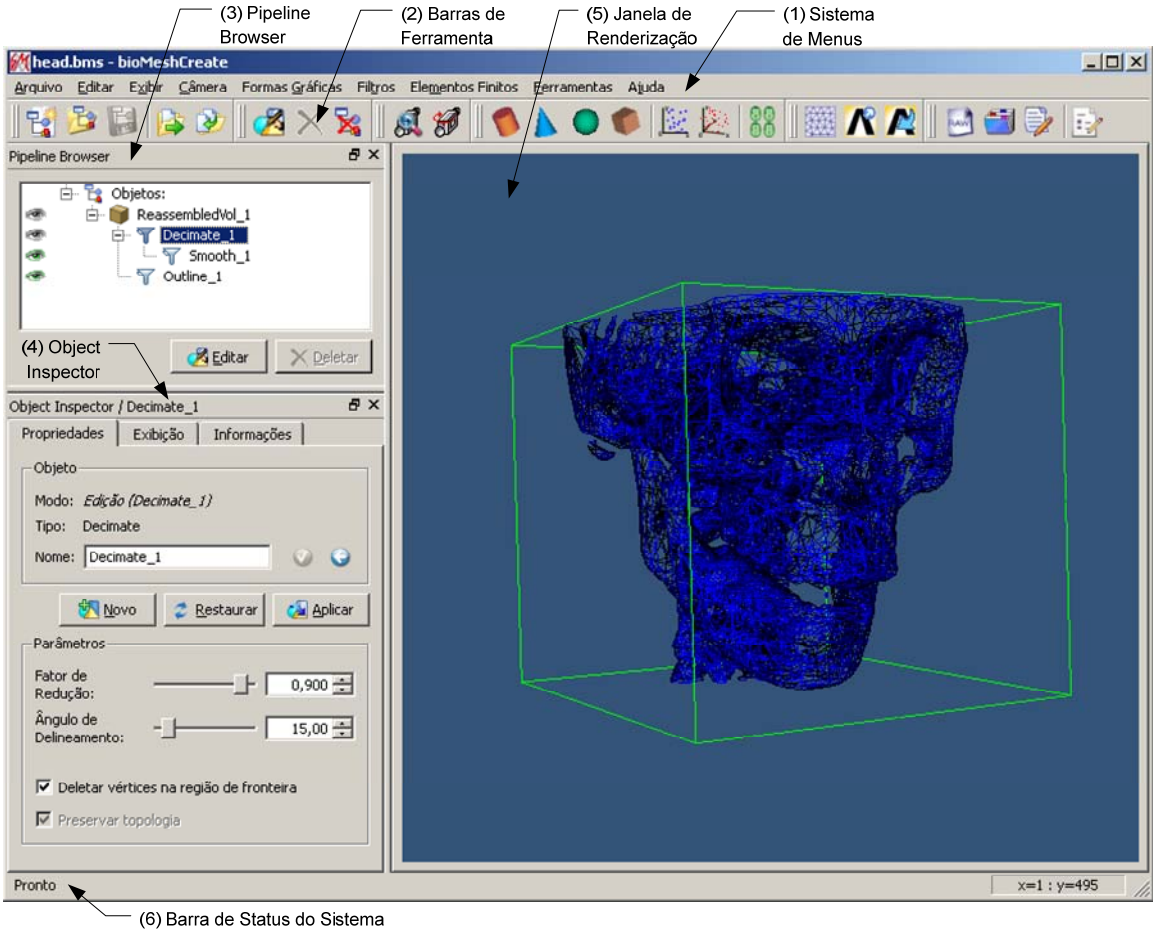


Figura 5.2 – Organização do programa bioMeshCreate. O crânio que aparece à direita foi reconstruído a partir de 93 fatias planas. / Fonte: O autor.

5.4 A FERRAMENTA DICOMCONVERTER

Toda a reconstrução 3D, a partir de imagens planas, executadas por bioMeshCreate, toma como entrada arquivos no formato RAW (seção 4.6.5), representando as fatias que irão

compor o volume. Mas, como imagens médicas originalmente se apresentam no formato DICOM, houve a necessidade de se construir uma ferramenta capaz de converter DICOM para RAW. Esta ferramenta, parte integrante do programa bioMeshCreate, foi desenvolvida e denominada *DicomConverter* (ou *dConv*).

A ferramenta, além de converter os arquivos DICOM para o formato RAW, também permite visualizar as seqüências de arquivos DICOM. O *Visualizador Dicom* possui uma barra de rolagem na parte inferior, pela qual é possível navegar pelas fatias carregadas. Na barra de título (parte superior) é mostrado o número da fatia exibida e o total de fatias.

A fatia mostrada na Figura 5.3 (lado direito) foi obtida de CT (Computed Tomography). Basicamente uma CT indica a quantidade de radiação absorvida por cada porção da seção analisada, e traduz essas variações numa escala de cinzas, produzindo uma imagem. Essa escala varia do preto (passagem do ar), tons de cinza (regiões menos densas) até o branco (região óssea).

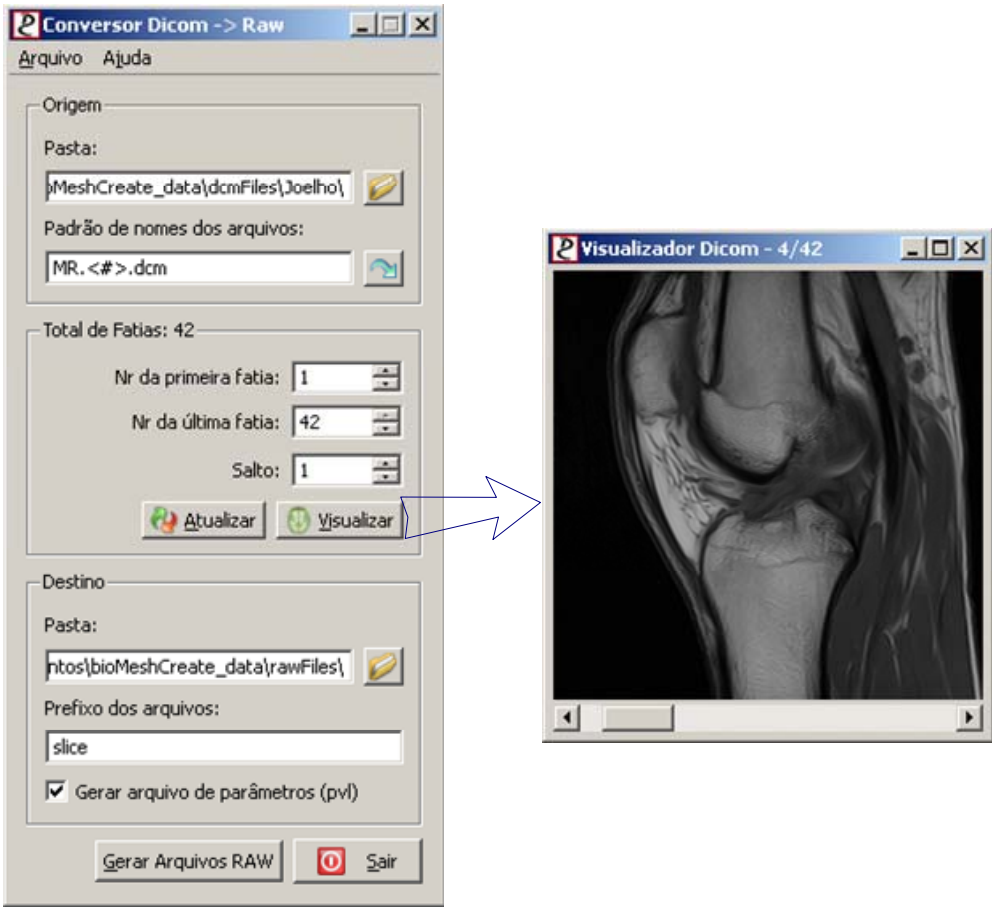


Figura 5.3 – A ferramenta DicomConverter exibindo uma fatia no formato DICOM. / Fonte: O autor.

Destaca-se neste formulário, a opção ‘Gerar arquivo de parâmetros (pvl)’, pois através desse arquivo no formato XML, fica mais fácil a reconstrução de volumes

posteriormente no bioMesh, uma vez que ele contém informações sobre a localização dos arquivos, número de fatias, tamanho, espaçamento entre fatias e extração de contornos. A Figura 5.4 ilustra um arquivo desse tipo. A extensão adotada para esse tipo de arquivo foi PVL, de parâmetros do volume.

```
<?xml version='1.0' encoding='iso-8859-1'?>
<rawfile>
  <source>
    <folder>C:\...\bioMeshCreate_data\rawFiles\</folder>
    <prefix>punho</prefix>
  </source>
  <slices>
    <first>1</first>
    <last>51</last>
  </slices>
  <size>
    <rows>512</rows>
    <cols>512</cols>
  </size>
  <spacing>
    <x>0,707</x>
    <y>0,707</y>
    <z>1,0</z>
  </spacing>
  <contour>
    <region>2</region>
    <filter>1300</filter>
    <normal>60,00</normal>
  </contour>
</rawfile>
```

Figura 5.4 – Exemplo de um arquivo de parâmetros de volume gerado com DicomConverter.
Fonte: O autor.

5.5 O MÓDULO DE RECONSTRUÇÃO DE VOLUMES

O primeiro passo para a reconstrução 3D com bioMeshCreate é converter o conjunto de imagens planas no formato DICOM para o formato RAW usando a ferramenta DicomConverter, conforme explicado anteriormente. Uma vez feita a conversão deve-se carregar o formulário de reconstrução de volumes no Object Inspector, e isto é feito através do menu *Formas Gráficas → Volume (reassembled)*.

O formulário responsável pela reconstrução de volumes pode ser observado na Figura 5.5. Este formulário admite preenchimento manual ou através da recuperação dos parâmetros presentes em um arquivo do tipo PVL que é gerado pela ferramenta de conversão, DicomConverter, no momento da conversão.

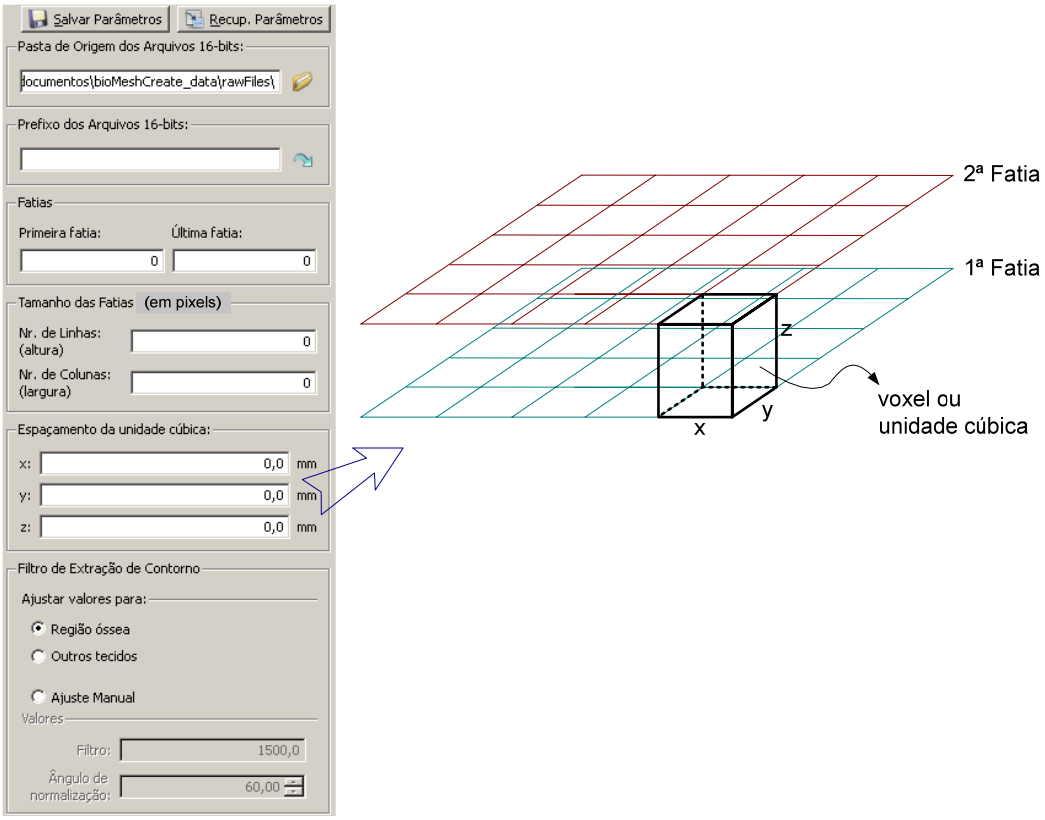


Figura 5.5 – Parâmetros de entrada para reconstrução de volumes. / Fonte: O autor.

Neste formulário, o tamanho das fatias corresponde à altura e à largura que a fatia ocupa, em pixels. Isto quer dizer que se uma fatia tem dimensões 64x64, então o tamanho do arquivo correspondente será:

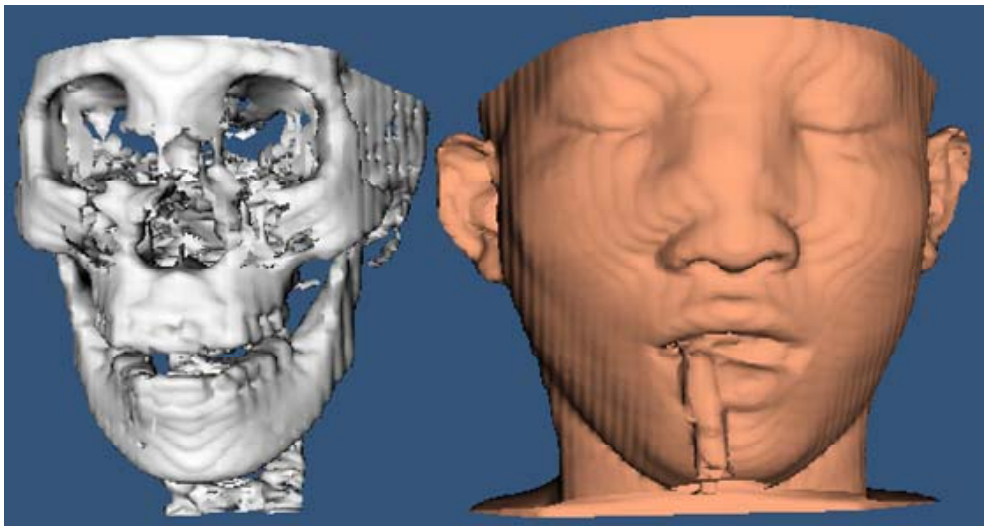
$$64 \,.\, 64 = 4096 \,.\, 2 = 8192 \text{ bytes ou, dividindo por } 1024 \Leftrightarrow 8 \text{ KB.}$$

A multiplicação por 2 se deve ao fato de que os arquivos RAW considerados, devem ser de 16-bits/pixel ou 2-bytes/pixel.

Quanto ao espaçamento da unidade cúbica, ele pode ser melhor entendido através da Figura 5.5 (lado direito), onde duas, das várias fatias usadas para compor um volume, são mostradas. O cubo que aparece em destaque (célula do tipo voxel) tem dimensões x, y, z e devem ser dadas em milímetros, correspondendo exatamente aos campos presentes nesta região do formulário. Note que as fatias apresentadas (grade) são formadas por pixels, que a partir de agora passam a ser dimensionadas por uma unidade do sistema métrico decimal, neste caso, milímetros.

Para a extração do contorno, o programa bioMesh toma as entradas: filtro e ângulo de normalização e implicitamente instancia um objeto da classe vtkContourFilter. Após vários testes, foi detectado que o valor 1500 para esse filtro foi o que melhor se adaptou à extração

de regiões ósseas, e o valor 500 para outros tecidos, como o epitelial. A Figura 5.6 mostra dois volumes reconstruídos a partir da mesma seqüência de fatias, porém com diferentes valores para o filtro de extração de contorno. Também é possível marcar a opção ajuste manual e modificar os valores do filtro de modo a atender situações específicas, ou fazer ajustes finos na imagem obtida.



(a) valor do filtro: 1500 (a) valor do filtro: 500

Figura 5.6 – Dois volumes reconstruídos com bioMeshCreate a partir de 93 fatias planas de 64x64 pixels.
Fonte: as imagens usadas na reconstrução fazem parte dos exemplos que acompanham a documentação do VTK.

Para compor um volume como o da Figura 5.6 foram necessárias 93 fatias de 64x64 pixels ou 8 KB, isto quer dizer que o volume final ocupa exatos 744 KB, sem levar em conta é claro, os recursos despendidos com filtros, renderização, iluminação da cena, entre vários outros. Agora, supondo que as fatias fossem imagens de 1 mega-pixel, o resultado final seria:

$$1000 \,.\, 1000 = 1000000 \,.\, 2 = 2000000 \text{ bytes} \cong 1953 \text{ KB} \cong 1{,}9 \text{ MB} \,.\, 93 \cong 177 \text{ MB}$$

Pelo cálculo acima, percebe-se que não deve ser levado em conta apenas o tamanho da fatia, mas também a quantidade de fatias envolvidas. Uma possível solução para esse aumento excessivo no consumo de memória se dá pela seleção da região de interesse (ROI – region of interest), diminuindo assim o tamanho de cada fatia antes da reconstrução.

O ângulo de normalização visa garantir uma boa superfície para a imagem. O algoritmo trabalha determinando a normal para cada polígono e fazendo uma média entre eles pelos pontos comuns (compartilhados). Quando arestas sinuosas (acentuadas) estão presentes, isto é, quando o ângulo médio calculado é maior que o valor informado, as arestas são quebradas, gerando novos pontos a fim de garantir a clareza dos contornos.

O processo de reconstrução 3D foi escrito usando objetos do VTK, sendo a superfície, extraída pelo algoritmo de *Marching Cubes*. Os detalhes desse algoritmo, bem como da implementação podem ser encontrados em Lorensen e Cline (1987) e também na documentação das classes do VTK (VTK DOCUMENTATION, 2007).

5.6 GERAÇÃO DE MALHA E TRANSFORMAÇÃO DE DADOS

No bioMeshCreate tanto a geração de malha como a maioria das transformações de dados são realizadas por objetos de processo do tipo filtro. Os filtros são objetos que requerem uma ou mais entradas de objetos de dados e, por sua vez, geram uma ou mais saídas. Os dados de saída normalmente são obtidos por operações realizadas nos dados de entradas, como a união, operações booleanas, transformações lineares, algoritmos de triangulação, redução poligonal, simplificação, e outras. Resumindo, os filtros diferem das formas gráficas basicamente pelo fato de precisarem de um objeto de dados como entrada, que pode ser uma forma gráfica ou mesmo outro filtro.

No bioMesh existem dez diferentes tipos de filtros. Os principais serão discutidos abaixo, sendo o primeiro deles, a Triangulação de Delaunay, empregado na geração de malhas.

5.6.1 Triangulação de Delaunay

Delaunay 2D. Parâmetros locais controlam a operação dos objetos de processo. O filtro Delaunay 2D, por exemplo, possui três parâmetros locais: fator de tolerância, fator alpha e fator de redução do elemento de malha (shrink).

O fator de tolerância é um parâmetro usado para determinar pontos coincidentes. Pontos localizados a uma distância menor que o valor de tolerância informado são considerados coincidentes, e um deles pode ser descartado durante a triangulação.

Usualmente a saída desse filtro é uma malha de triângulos, mas se o valor da distância alpha for diferente de zero, então triângulos, arestas e vértices com raio alpha serão renderizados. Em outras palavras, alpha diferente de zero resulta em uma combinação

arbitrária de triângulos, linhas e vértices. Maiores detalhes sobre o valor alpha podem ser encontrados no trabalho de Edelsbrunner e Mücke (1994).

Em duas dimensões a triangulação de Delaunay tem mostrado ser ótima, ou seja, dada uma nuvem de pontos, esta triangulação é a que resulta em um conjunto de triângulos o mais próximo possível de triângulos eqüiláteros. No bioMeshCreate um teste pode ser feito renderizando um conjunto de pontos aleatórios no plano, e aplicando a triangulação de Delaunay 2D com fator de tolerância igual a zero. A Figura 5.7 seguinte ilustra essa situação. Note que na malha de triângulos gerada, todos os pontos do conjunto tornaram-se vértices de algum triângulo (Figura 5.7b).

O fator de redução de elemento tem por objetivo, facilitar a visualização dos elementos gerados. Quando esse fator é zero significa sem redução (Figura 5.7a) e, neste caso, os elementos adjacentes ficam justapostos preenchendo toda a região dos pontos.

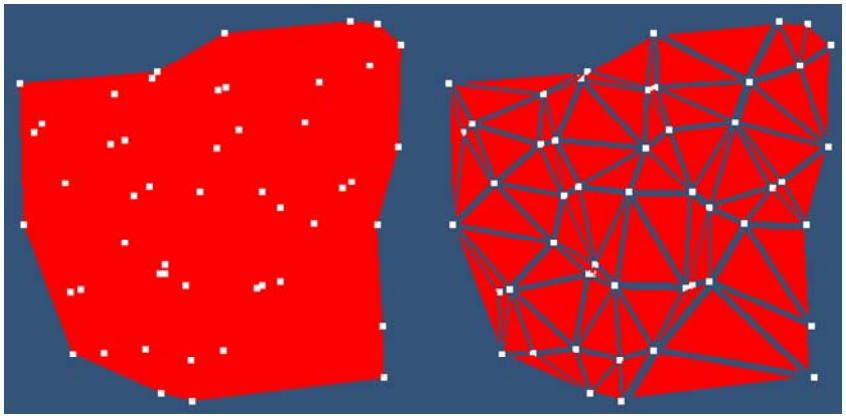


Figura 5.7 – Malha de triângulos gerada a partir de 50 pontos aleatórios no plano-xy:
(a) sem redução do elemento de malha e (b) com redução igual a 0,8. / Fonte: O autor.

Delaunay 3D. O filtro de Delaunay 3D, quanto aos parâmetros de entrada, é idêntico ao 2D, ou seja, possui fatores de tolerância, alpha e redução, que funcionam do mesmo modo. A triangulação de Delaunay 3D, no entanto, não é ótima, podendo para certos conjuntos de pontos, resultarem elementos 3D degenerados.

A Figura 5.8 mostra a triangulação de Delaunay 3D aplicada sobre uma viga. O resultado é uma malha de volume constituída por elementos tetraédricos.

A triangulação de Delaunay 3D é um método bastante usado para a geração de malhas volumétricas, porém, com algumas restrições, principalmente no que se refere à sua aplicação sobre regiões não-convexas.

Mais detalhes sobre a triangulação de Delaunay podem ser encontrados na seção 3.2.

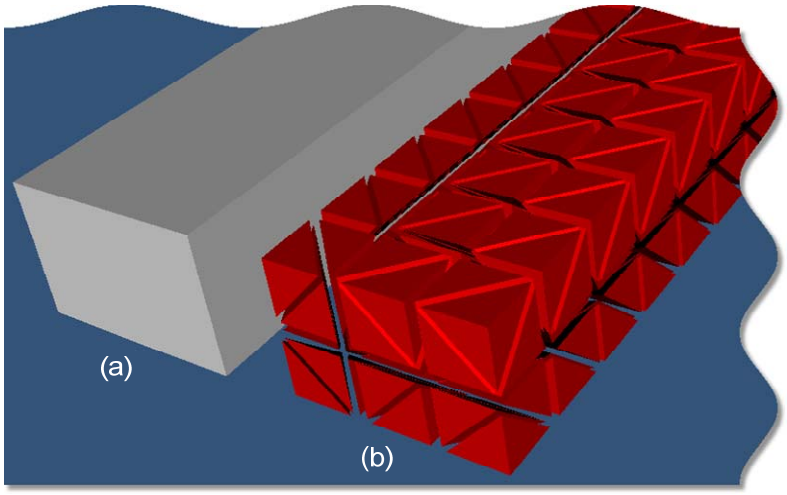


Figura 5.8 – (a) Viga gerada com elemento cúbico de dimensões x = 60, y = 40 e z = 400. (b) triangulação de Delaunay 3D aplicada sobre a viga, com tolerância = 0,1; alpha = 0,0 e redução = 0,8. / Fonte: O autor.

5.6.2 Geração de Malha Triangular

Este filtro não possui parâmetros locais, ele simplesmente recebe uma entrada de dados poligonais e quebra os polígonos em triângulos. A Figura 5.9 é um típico exemplo, onde os polígonos que formam a viga (Figura 5.9a) são transformados em triângulos (Figura 5.9b). Uma vez gerada a malha de triângulos é possível exportar o objeto para o formato STL. O formato STL é empregado em prototipagem rápida e aceito por muitos sistemas hoje em dia, facilitando assim a comunicação de dados entre diferentes aplicações.

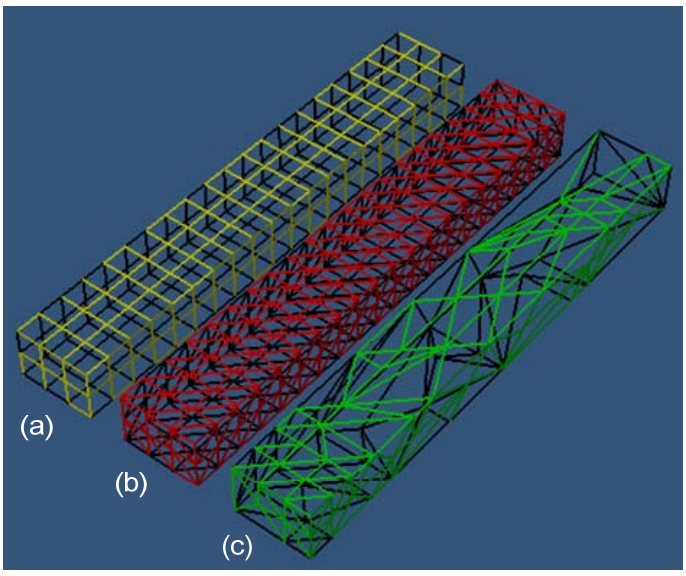


Figura 5.9 – (a) Viga constituída por malha poligonal; (b) malha triangular gerada sobre a viga; (c) simplificação de 60% da malha triangular gerada. / Fonte: O autor.

5.6.3 Simplificação de Malha Triangular

Este filtro reduz o número de triângulos em uma malha (decimation), gerando uma boa aproximação da geometria original. Um exemplo de sua aplicação pode ser visto na Figura 5.9c.

O filtro de simplificação de malha triangular possui quatro parâmetros locais: 1) fator de redução; 2) ângulo de delineamento; 3) exclusão de vértices de fronteira e 4) preservar topologia. Sendo este último, um parâmetro fixo nesta versão, pois se observou durante os testes que a divergência topológica, na maioria das vezes, causava degenerações nos conjuntos de dados obtidos.

O fator de redução é uma taxa que varia de 0 a 1, e indica o percentual de redução da malha.

O ângulo da normal à superfície entre dois triângulos adjacentes corresponde ao ângulo de delineamento especificado (Figura 5.10a).

Um simples ponto pode ser classificado como um ponto interior ou um ponto de fronteira. Esta classificação é baseada na geometria local da malha. É possível determinar se o programa deve considerar vértices na região fronteiriça como um candidato à deleção ou não.

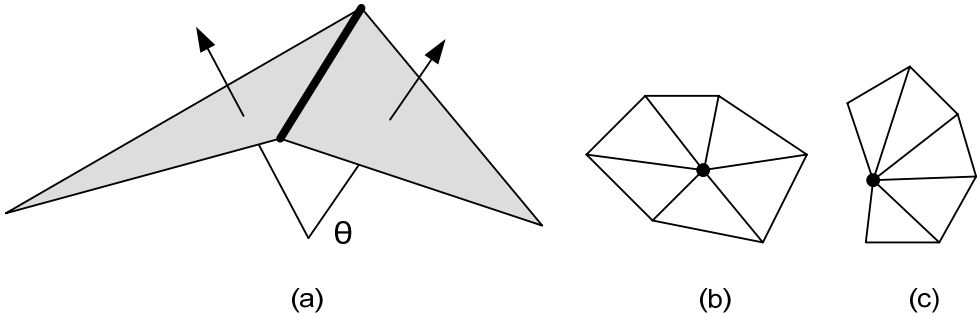


Figura 5.10 – (a) Ângulo de delineamento; (b) ponto interior; (c) ponto de fronteira.
Fonte: Adaptado de Schroeder, Martin e Lorensen (2006).

5.6.4 Suavização de Superfícies

Suavização de superfície ou alisamento da malha (smoothing) é uma técnica que ajusta as coordenadas dos pontos de um conjunto de dados. O propósito da suavização é melhorar a aparência da malha. Durante a suavização a topologia do conjunto de dados não é modificada, somente a geometria. O efeito é então “relaxar” a malha, fazendo células com

melhor aparência e vértices eventualmente mais distribuídos. Este filtro possui seis parâmetros locais: 1) número de iterações; 2) fator de convergência; 3) ângulo de suavização; 4) fator de relaxação; 5) suavização de vértices de fronteira e 6) diferenciação entre vértices simples, internos e fixos.

De maneira resumida o algoritmo procede da seguinte maneira: para cada vértice **v**, uma análise topológica e geométrica é executada para determinar quais vértices e quais células estão conectadas a **v**. Então uma matriz de conectividade é construída para cada vértice. A seguir, uma fase de iteração inicia-se sobre todos os vértices. Para cada vértice **v**, as coordenadas de **v** são modificadas de acordo com a média dos vértices conectados. Um fator de relaxação é avaliado para controlar a quantidade de deslocamentos de **v** (fator percentual variando de 0 a 1). O processo repete-se para cada vértice. Este passo sobre a lista de vértices é uma simples iteração. Muitas iterações (geralmente cerca de 20 ou mais) devem ser executadas a fim de chegar a um bom resultado. A opção de suavizar vértices de fronteira (Boundary Smoothing) habilita/desabilita a operação de suavização em vértices que estão na fronteira da malha. Um vértice de fronteira é aquele que está envolvido por um semi-ciclo de polígonos ou usado por uma simples linha (Figura 5.10c)

Outro parâmetro importante é a opção que permite diferenciar vértices simples, internos e fixos quando o algoritmo é executado.

A convergência é o limite máximo no movimento do ponto. Se o máximo movimento durante uma iteração é menor do que a convergência, então o processo de suavização termina. Convergência é expressa como uma fração da diagonal da caixa de contorno da imagem (outline).

O ângulo de suavização segue o mesmo critério utilizado pelo filtro de simplificação de malha, ou seja, é o ângulo da normal à superfície entre dois triângulos adjacentes (Figura 5.10a).

5.7 O MÓDULO DE ELEMENTOS FINITOS

O Módulo de Elementos Finitos preocupa-se com a integração dos modelos gerados pelo bioMeshCreate com um software de análise por elementos finitos. Neste caso foi utilizado o software Ansys Multiphysics (2007). Este módulo é constituído por: 1) Módulo de Exportação; 2) Módulo de Integração; e 3) Ferramenta de Controle.

5.7.1 Ansys – Módulo de Exportação

O *Módulo de Exportação* toma como entrada uma triangulação de Delaunay 3D e como saída gera dois arquivos: um de nós e outro de elementos, compatíveis com o formato do Ansys. Um arquivo de LOG normalmente também é gerado ao final do processo de exportação, contendo informações sobre o modelo exportado.

A Figura 5.11, abaixo, ilustra a interface gráfica do Módulo de Exportação. Neste formulário deve-se escolher a triangulação de Delaunay 3D que será exportada, caso exista mais de uma renderizada; a pasta de destino para onde serão gerados os arquivos; e o prefixo do nome dos arquivos. Também é possível tecer alguns comentários sobre o modelo (opcional).

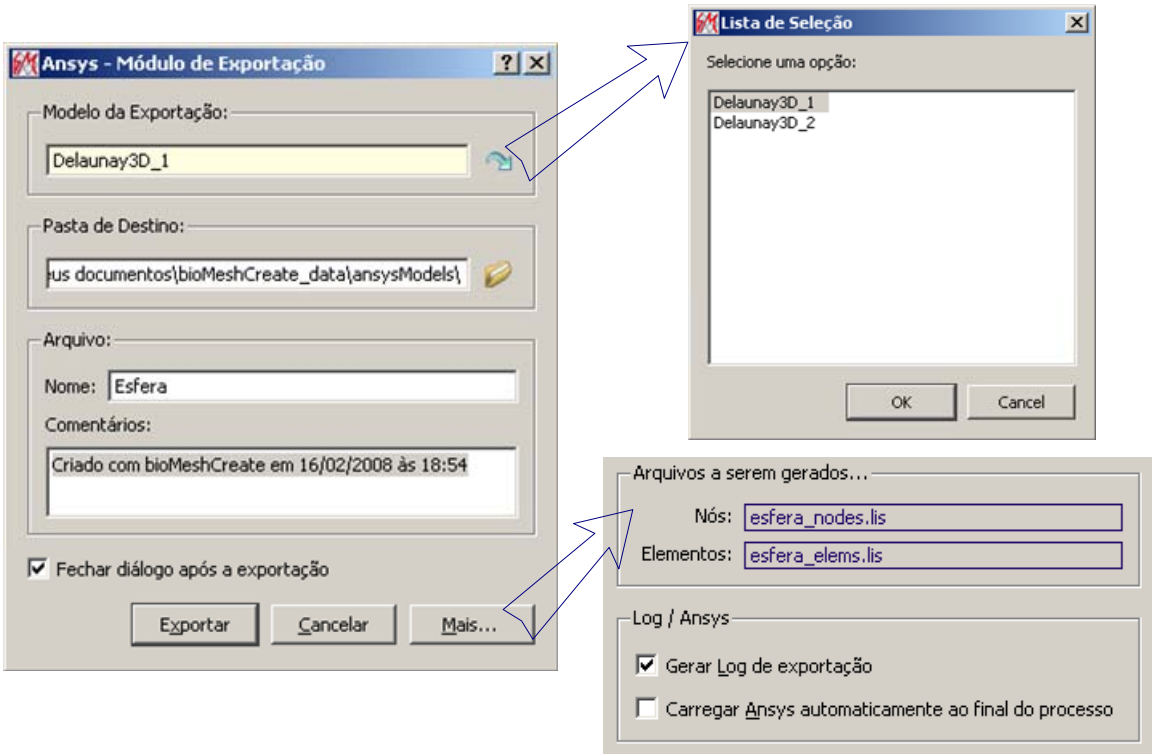


Figura 5.11 – Interface gráfica do Módulo de Exportação de Dados para o Ansys. / Fonte: O autor.

Ao final do processo de exportação, dois arquivos com extensão LIS (de listagem) serão gerados, contendo os nós (coordenadas) e os elementos (incidência nodal) do modelo exportado.

Nas opções adicionais ainda é possível definir se o programa deve gerar o arquivo de LOG ao final da exportação, e se o Ansys deve ser carregado imediatamente após o término do processo.

5.7.2 Ansys – Módulo de Integração

A integração com o Ansys é realizada através de scripts ou macro-comandos desenvolvidos em APDL (Ansys Parametric Design Language) que são instalados no diretório de trabalho do Ansys automaticamente, quando o bioMeshCreate é configurado. Para detalhes sobre as configurações do bioMeshCreate, ver Jóia Filho (2008) – Capítulo 4. Sobre a APDL, informações podem ser obtidas em Ansys (2005a).

Depois de configurado, o *Módulo de Integração*, irá carregar o programa Ansys e com ele o botão BIO_MESH em sua barra de ferramentas, conforme ilustrado na Figura 5.12. Este botão, por sua vez, é o responsável pela carga da Ferramenta de Controle e importação dos modelos.

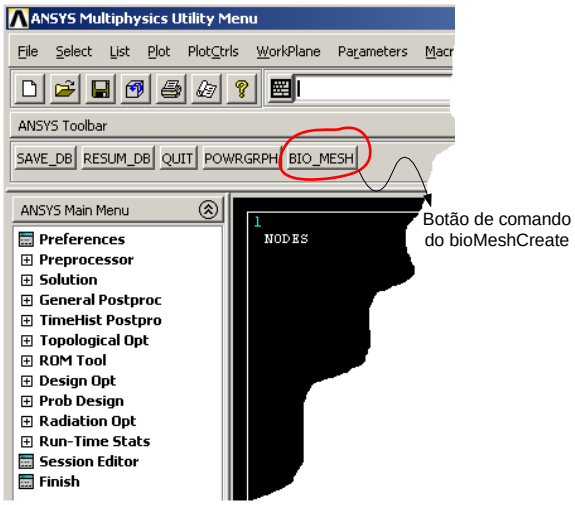


Figura 5.12 – Barra de ferramentas do Ansys modificada por macro-comandos. / Fonte: O autor.

Na verdade, as ações desse módulo são executadas em três etapas. Assim, três scripts APDL foram desenvolvidos: 1) mcTlbar.mac; 2) mcSelMod.mac; e 3) mcPrep7.mac.

O primeiro script, *mcTlbar.mac*, é responsável pela criação do botão de comando BIO_MESH na barra de ferramentas do Ansys. Seu código é simples, conforme listado abaixo:

```
1 /NOPR
2 *ABB,BIO_MESH,MCSELMOD
3 /GO
```

O segundo script de comandos APDL, *mcSelMod.mac*, é o responsável pelo carregamento da Ferramenta de Controle. Quando o botão BIO_MESH é clicado, esta ferramenta é carregada em um processo separado e o controle do Ansys é transferido para este

novo processo, permanecendo até que seja encerramento. O comando que realiza esta ação no Ansys, isto é, que passa uma cadeia de comando e seus argumentos para o sistema operacional é o /SYP (System Process). A listagem parcial desse script é mostrada abaixo, estando o código completo, disponível no apêndice B.

```
 1 /NOPR
 2 /syp, C:\Arquiv~1\bioMes~1\ansysCtrl.exe, 'enter_ansys_mode', 'true'
 3 fParams =
 4
 5 *dim,fParams,array,7
 6 *vread,fParams(1),'mcArgs','lis',,,1,,,2
 7 (7F6.0)
 8
...                                    ...
26 /GOPR
```

A troca de parâmetros entre as duas aplicações é realizada por arquivo. Isto significa que quando a Ferramenta de Controle é fechada, um novo arquivo contendo os parâmetros informados é automaticamente gerado (mcArgs.lis). Nesse ponto, o Ansys assume de volta o controle da aplicação e o script continua a ser executado a partir da linha 3. Então o arquivo de parâmetros previamente gerado é carregado e seus valores devidamente alocados em variáveis. Por fim, o script mcPrep7 é chamado para completar o processo de integração dos modelos gerados com bioMeshCreate.

O script *mcPrep7.mac* inicia o modo de pré-processamento do Ansys, carrega os nós e os elementos a partir dos arquivos LIS, define *keypoints*, e cria áreas e volumes, se definido no arquivo de parâmetros. O código desse script, por ser extenso, encontra-se disponível no apêndice B.

5.7.3 A Ferramenta de Controle: AnsysControl

Quando um modelo é gerado através do Módulo de Exportação, uma base de informações XML é atualizada, contendo dados sobre o modelo gerado (metadados). A Ferramenta de Controle, denominada *AnsysControl* (ou *AnsysCtrl*) é responsável por exibir esses modelos em uma lista de seleção, conforme pode ser observado na Figura 5.13.

Através dessa ferramenta é possível visualizar os arquivos correspondentes ao modelo (nós, elementos e LOG), gerenciar os modelos existentes (adicionar, editar e excluir), e modificar alguns parâmetros (botão Mais...). Para mais detalhes, ver Jóia Filho (2008).

Uma vez selecionado o modelo e confirmada a importação (botão OK), o arquivo de parâmetros *mcArgs.lis* é gerado a fim de ser processado pelo script de comandos APDL, mcPrep7.mac.

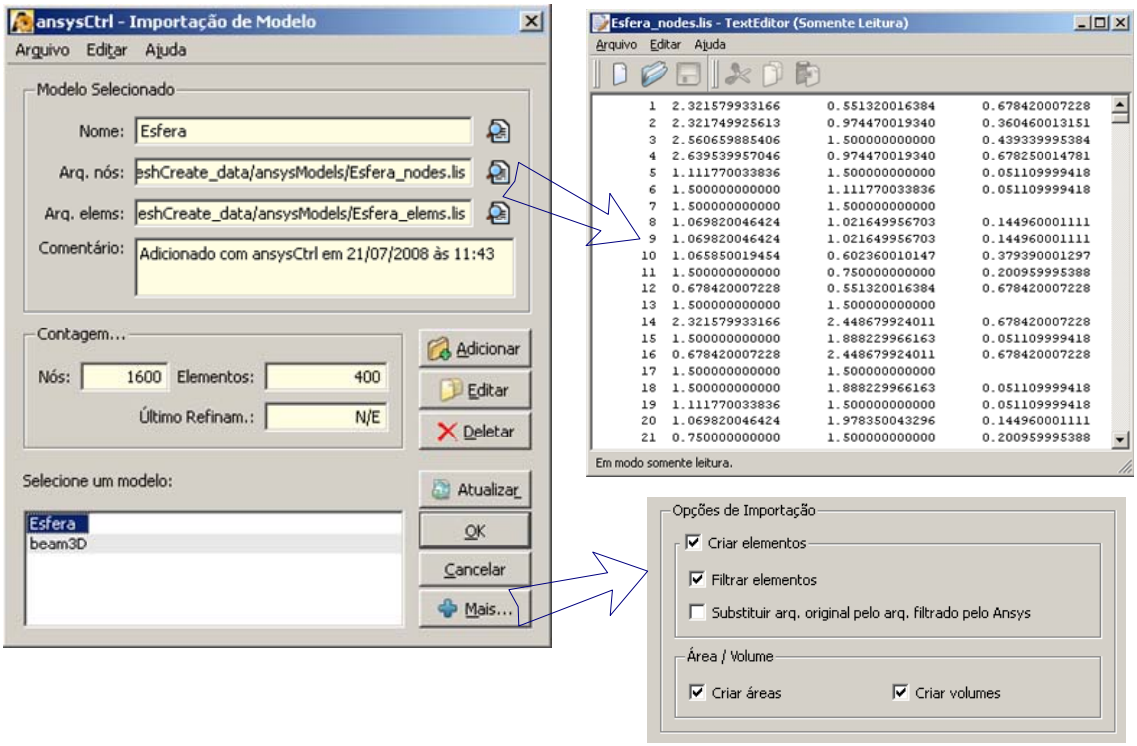


Figura 5.13 – Interface gráfica da ferramenta AnsysControl. / Fonte: O autor.

A Figura 5.14 representa um modelo esférico gerado com bioMeshCreate, e importado para o Ansys 10 usando o Módulo de Elementos Finitos.

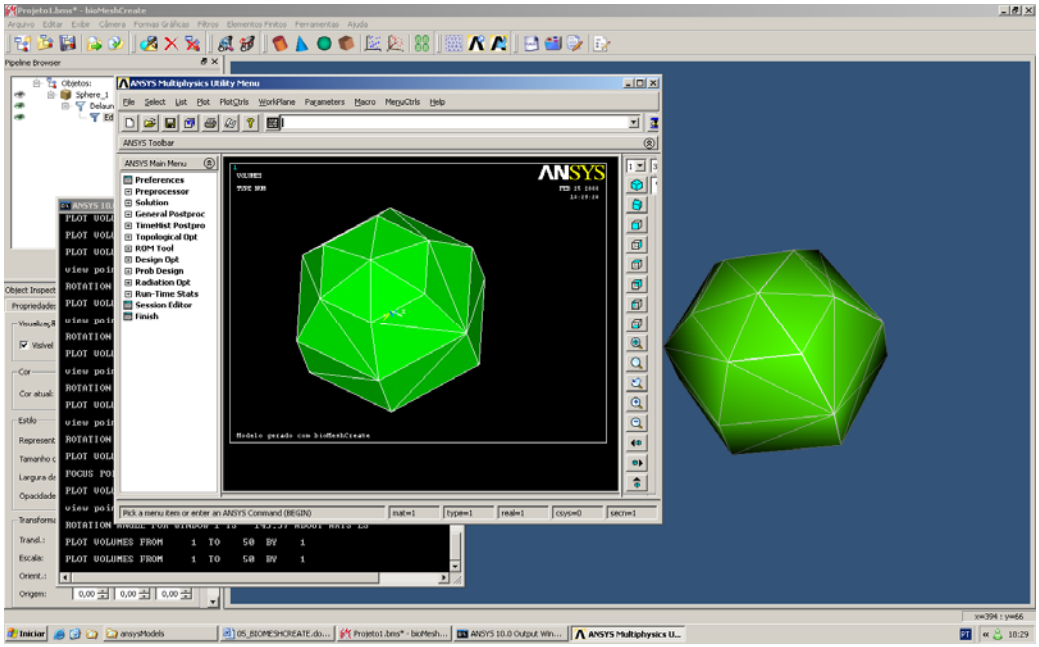


Figura 5.14 – Triangulação de Delaunay3D aplicada sobre um modelo esférico e importado pelo Ansys, através da ferramenta AnsysCtrl. / Fonte: O autor.

6 APLICAÇÕES A ESTRUTURAS BIOMECÂNICAS

O programa bioMeshCreate, conforme já mencionado, foi desenvolvido com o intuito de resolver problemas aplicados à biomecânica, relacionados à fase que precede a análise por elementos finitos, ou seja, obtenção do modelo geométrico (região óssea) para posterior análise em softwares especializados. Esta solução sistematizada envolve várias etapas que serão discutidas nas próximas seções. Assim, a seção 6.1 exibe os passos necessários para a obtenção do modelo geométrico, de maneira geral, através de diagramas de fluxo, e isto inclui tanto o processo da reconstrução 3D como o de geração da malha. A seção 6.2 concentra-se somente na parte relacionada à reconstrução 3D, através de um caso de uso (exemplo prático). Paulatinamente, através desse exemplo, a seção 6.3 discute o processo de geração da malha de volume pelo programa bioMeshCreate, e na seção 6.4 pelo próprio software de análise por elementos finitos. Finalizando, a seção 6.5 aponta as limitações e vantagens de cada processo.

6.1 AS ETAPAS DE OBTENÇÃO DO MODELO

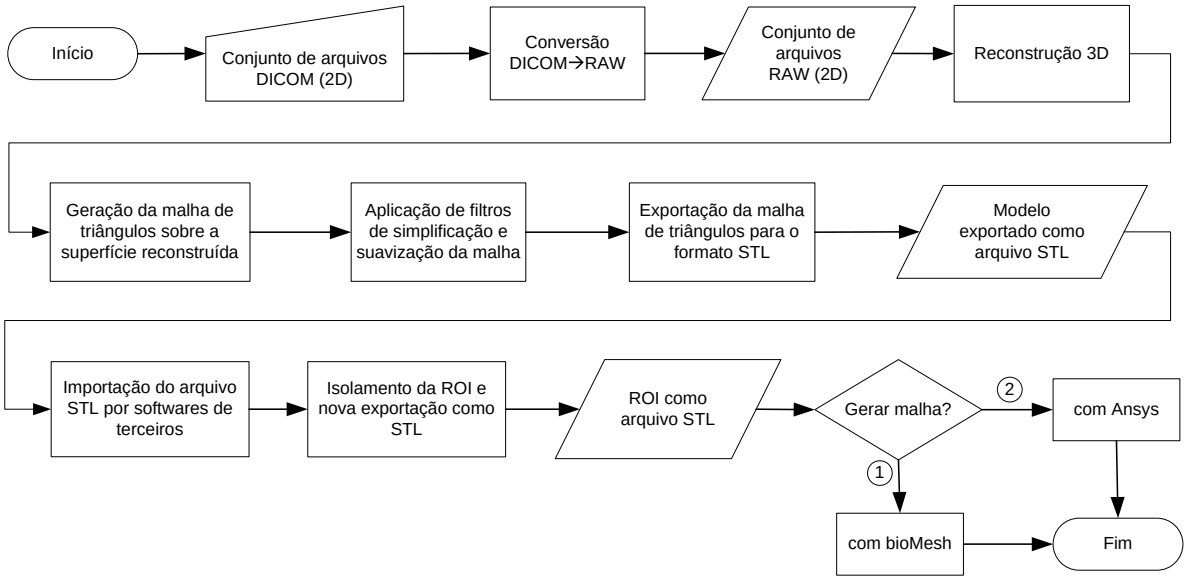


Figura 6.1 – Fluxograma da reconstrução 3D.
Fonte: O autor.

O fluxograma da Figura 6.1 contempla todo o processo da reconstrução tridimensional e obtenção do modelo geométrico para posterior análise por elementos finitos. No entanto, o processo de geração da malha, tanto pelo bioMesh quanto pelo Ansys, foram subdivididos em outros dois fluxogramas para facilitar o estudo, e podem ser observados pelas Figuras 6.2 e 6.3, respectivamente.

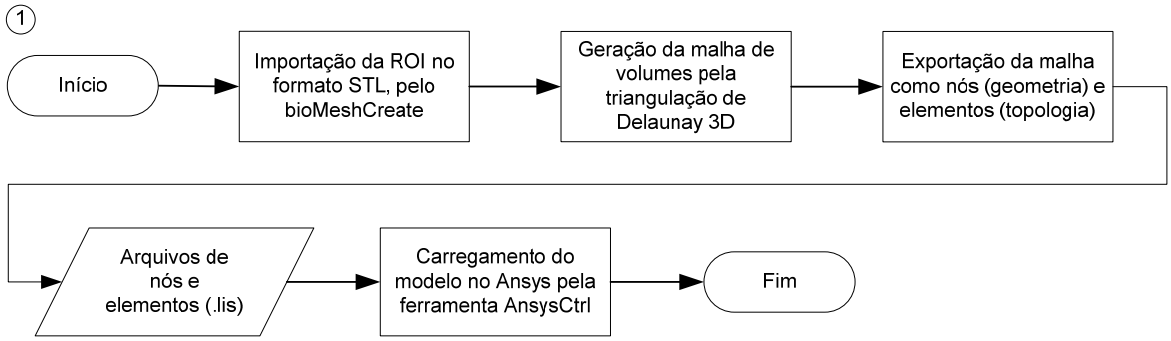


Figura 6.2 – Fluxograma da geração de malha com bioMeshCreate.
Fonte: O autor.

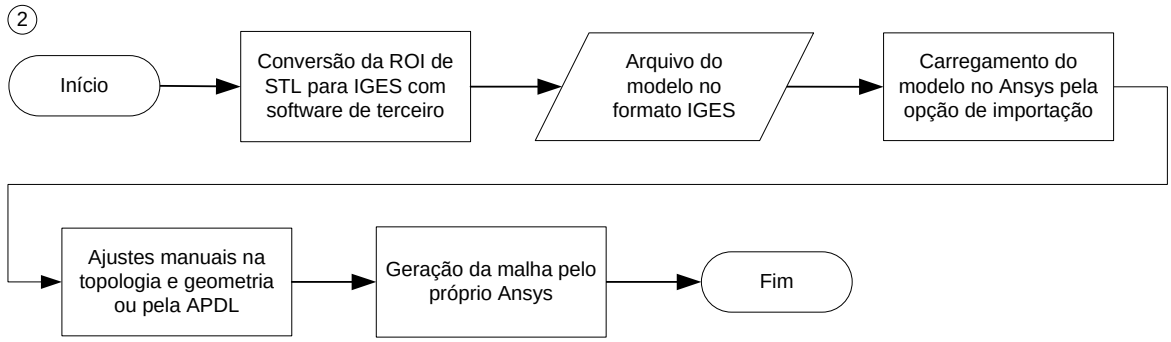


Figura 6.3 – Fluxograma da geração de malha com o Ansys.
Fonte: O autor.

Estes três fluxogramas encerram, portanto, todo o processo de obtenção do modelo computacional e serão explicados nas próximas seções.

6.2 O PROCESSO DE RECONSTRUÇÃO

O processo de reconstrução tridimensional envolve os passos presentes no fluxograma da Figura 6.1 até a obtenção da ROI (Region Of Interest) como arquivo STL. Para isto considere a reconstrução de um punho humano direito, a partir de 51 fatias planas no formato DICOM. Cabe destacar que estas imagens representam dados reais de um paciente

que apresentava sintomas de LER (Lesão por Esforços Repetitivos), e estão presentes no CD de instalação do bioMeshCreate, localizadas sob o caminho:

\bioMeshCreate_data\dcmFiles\Punho.

Os arquivos são provenientes de CT. A Figura 6.4 mostra algumas dessas fatias visualizadas com a ferramenta DicomConverter.

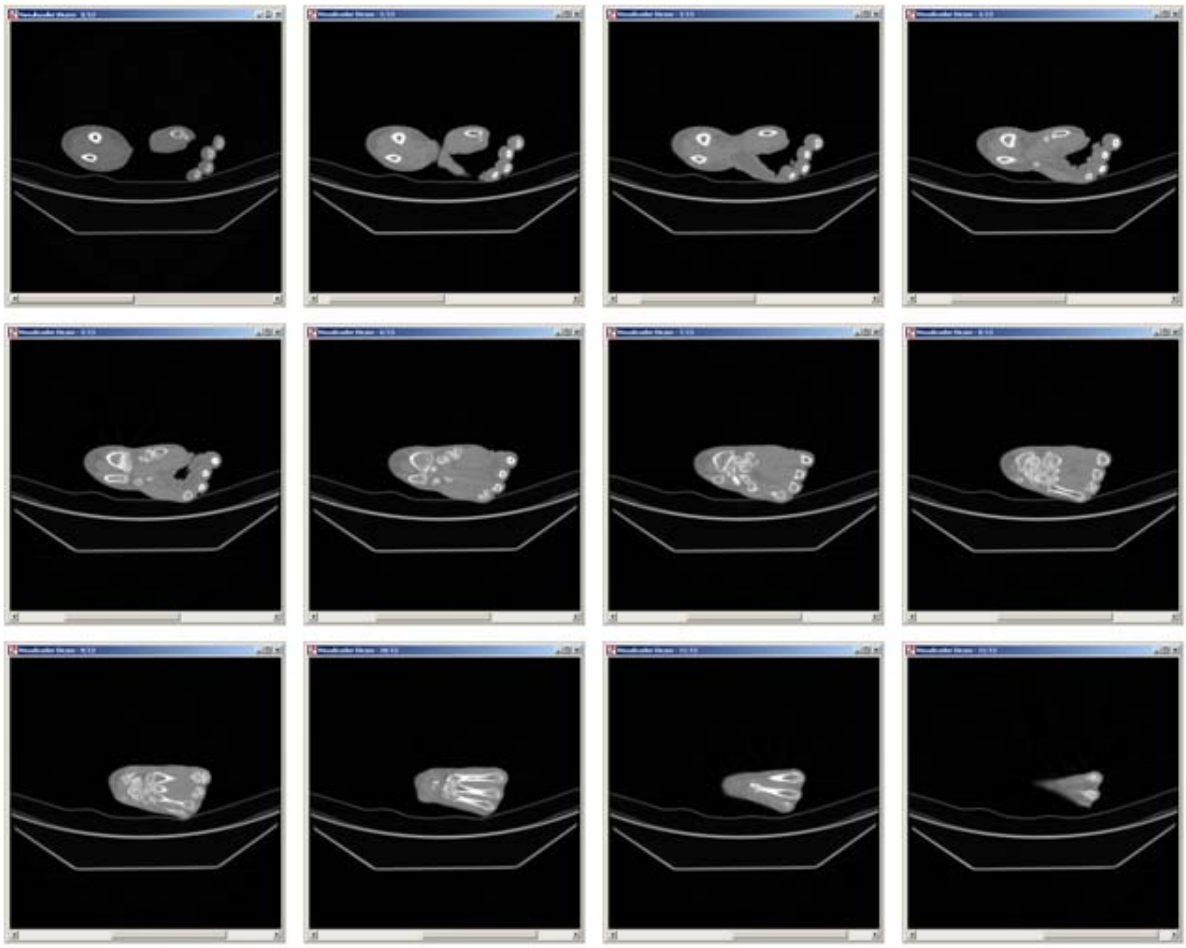


Figura 6.4 – Imagens de CT do punho humano, visualizadas com DicomConverter.
Fonte: O autor.

Uma vez que se têm os dados de entrada, isto é, a seqüência de arquivos DICOM, o primeiro passo é realizar sua conversão para o formato RAW, através da ferramenta DicomConverter. Depois disso, com os arquivos no formato RAW, deve-se carregar o módulo de reconstrução de volumes no Object Inspector e executar a reconstrução.

A Figura 6.5 representa o punho reconstruído. À esquerda (região óssea) utilizou-se o valor 1300 para o filtro de extração de contornos e à direita (pele) foi utilizado o valor 500. Neste caso, o mesmo conjunto de fatias pode reconstruir diferentes regiões, dependendo do valor adotado para o filtro. Cabe lembrar, que tais valores são arbitrários e obtidos após vários testes com a estrutura, na busca do melhor resultado.

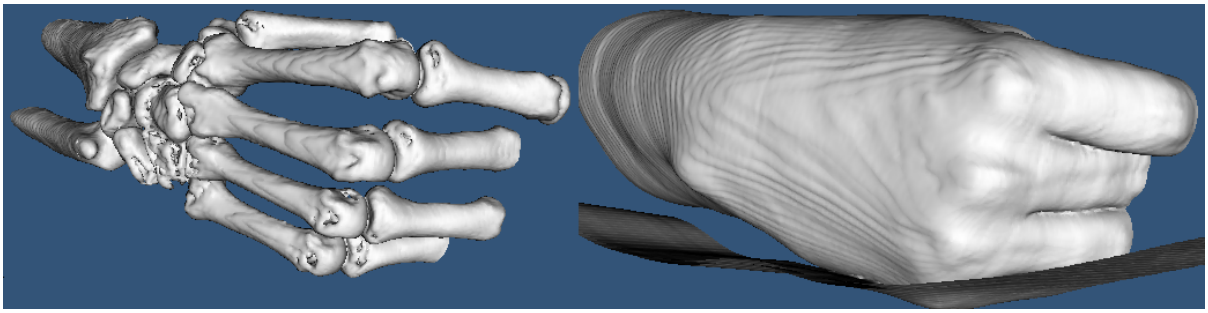


Figura 6.5 – Reconstrução da região do punho com bioMehsCreate, a partir de 51 fatias planas.
Fonte: O autor.

A malha de triângulos é obtida por um filtro de extração de isosuperfícies, logo após o processo de reconstrução 3D. Sua visualização pode ser feita através do Object Inspector, na guia exibição, mudando o atributo de representação do objeto, de superfície para linhas. Como exemplo, a Figura 6.6 representa a mesma região óssea obtida na Figura 6.5, porém mais ampliada e mudando a forma de representação para linhas.

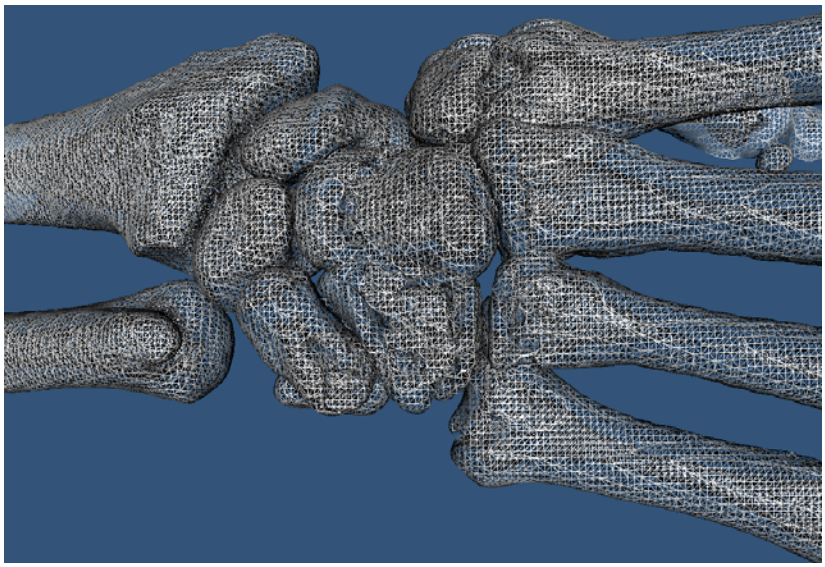


Figura 6.6 – Região óssea ampliada várias vezes mostrando a malha de triângulos na superfície, neste caso, 496.280 triângulos – 15,2 MB. / Fonte: O autor.

A aplicação de filtros ajuda a diminuir a resolução das imagens e melhorar o resultado. Na Figura 6.7 foram aplicados dois filtros sobre a estrutura inicial. O primeiro, um filtro de simplificação da malha triangular, reduziu severamente o número de triângulos e a quantidade de memória necessária para seu armazenamento. O segundo, um filtro de suavização de superfícies, ajustou a posição dos pontos a fim de melhorar a visualização.

Pelo elevado número de triângulos existentes (Figura 6.6), certos filtros levaram mais tempo para serem processados. Alguns testes foram realizados nesse sentido e os resultados serão apresentados a seguir.

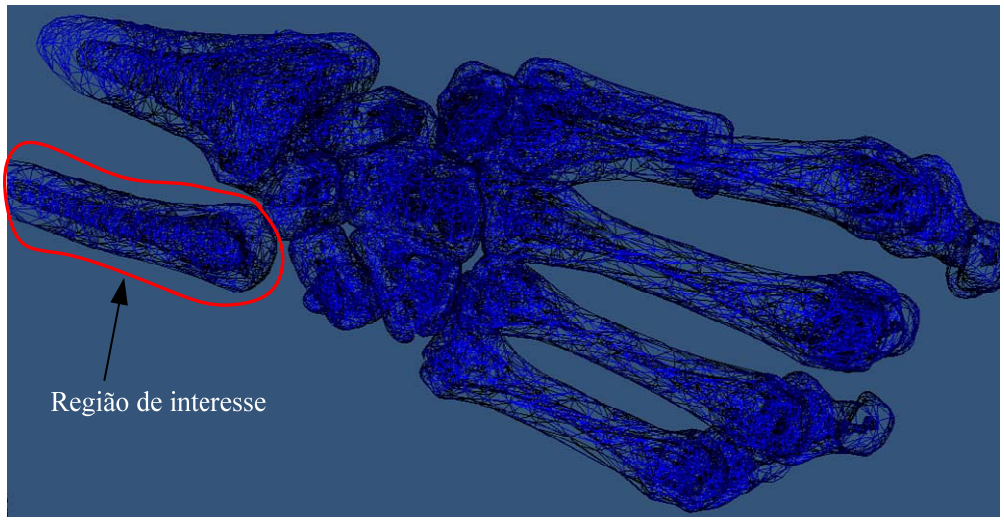


Figura 6.7 – Malha de triângulos simplificada 90% pelo filtro de redução e depois suavizada pelo filtro de alisamento através de 20 repetições sucessivas. Conta agora com 85.166 triângulos e 3,36 MB. / Fonte: O autor.

Para chegar ao resultado mostrado na Figura 6.7, partindo das imagens já convertidas para o formato RAW, foram necessários basicamente três passos: 1) reconstrução do volume; 2) simplificação da malha; e 3) suavização da superfície. Uma avaliação do tempo de processamento/renderização de cada etapa foi realizada em dois equipamentos diferentes. A descrição dos equipamentos, bem como os resultados obtidos em cada um deles, seguem abaixo:

Tabela 1 – Descrição dos equipamentos utilizados no processo de reconstrução.

Equipamento	Descrição
Máquina 1	• AMD Athlon – 64 X2 com Processador Dual Core 5000+ 2.61 GHz; • 4,00 GB de Memória RAM; • Placa de vídeo com acelerador gráfico NVIDIA GeForce 8600 GT / 256 MB de memória.
Máquina 2	• Pentium IV – CPU 2.4 GHz; • 1,00 GB de memória RAM; • Placa de vídeo SVGA comum – memória compartilhada.

Tabela 2 – Tempo de processamento das etapas da reconstrução.

Processamento Realizado / Tempo (em segundos)	Máquina 1	Máquina 2
Reconstrução do Volume	06	11
Simplificação da Malha	21	34
Suavização da Superfície	02	03
Tempo total de processamento (s)	*29*	*48*

Na Tabela 1 foram listadas as características principais de cada equipamento. Supostamente, aquelas que podem influenciar diretamente no desempenho de sistemas gráficos desta natureza: o processador, a quantidade de memória disponível, e o adaptador de vídeo instalado.

Na Tabela 2 foi levantado o tempo de processamento de cada etapa, mediante implementação direta de um *timer*, em cada rotina computacional. O aumento total do tempo de execução, do primeiro para o segundo equipamento, ficou em torno de 65%, o que pode ser justificado pela evidente diferença de capacidade entre os mesmos.

Existem vários outros fatores que podem interferir nos resultados obtidos, tais como o sistema operacional instalado, os serviços, e os processos em execução no momento do cálculo. Entretanto, o objetivo desta análise foi apenas fornecer um parâmetro de referência do tempo gasto na execução de cada etapa, em diferentes equipamentos.

O último passo é isolar a região de interesse que aparece em destaque na Figura 6.7. O programa bioMeshCreate ainda não possui recursos para isolamento da região de interesse. Para completar o processo, portanto, foi utilizado o software Rhinoceros (2007). A comunicação de dados entre os dois sistemas foi feita através da exportação do modelo reconstruído e filtrado pelo bioMeshCreate para o formato de malha de triângulos, o STL.

Uma vez gerado o arquivo STL através do módulo de exportação, ele pôde ser carregado pelo programa Rhinoceros (ou Rhino). Daí foi possível, a partir de sua interface gráfica, limpar a imagem de ruídos (partes indesejáveis) e isolar a região de interesse (Figura 6.8).

O Rhinoceros também oferece alguns filtros para melhorar a qualidade da malha. Um deles é a capacidade de agrupar (collapse) pequenas áreas automaticamente, baseado no valor da menor área informada. Também é capaz de fechar buracos (holes) existentes na malha, mesclando contornos abertos ou segmentos de linha para áreas adjacentes a fim de produzir volumes fechados. Essas operações também podem ser realizadas pelos softwares de análise por elementos finitos, na fase de pré-processamento.

Estando a região de interesse devidamente selecionada, ela pode ser salva novamente como um arquivo STL. A partir daí inicia-se o processo de geração da malha de volume. Dois estudos foram realizados nesse sentido, e serão discutidos nas próximas seções. No primeiro, mais automatizado, a malha é gerada e exportada pelo bioMeshCreate, embora com algumas restrições. No segundo, mais interativo, porém mais eficiente, a malha é gerada pelo próprio software de análise por elementos finitos, neste caso, o Ansys.

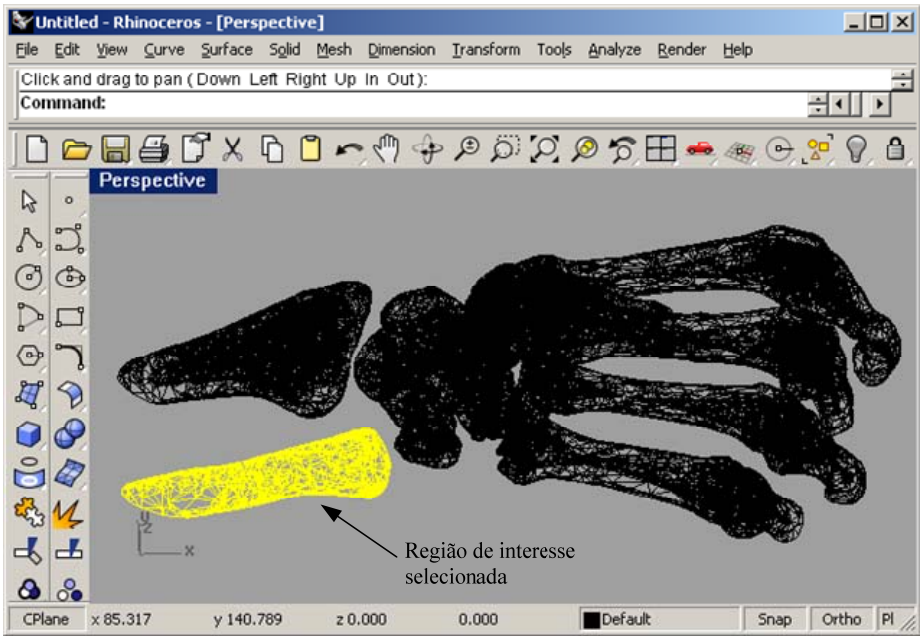


Figura 6.8 – Região de interesse selecionada a partir do arquivo STL, com o uso do Rhinoceros 4.0.
Fonte: O autor.

6.3 O PROCESSO DE GERAÇÃO DE MALHA PELO BIOMESH

Este processo segue o fluxograma da Figura 6.2, logo o primeiro passo a ser dado é a importação da ROI, no formato STL, pelo bioMeshCreate, através do módulo de exportação / importação. A Figura 6.9 mostra a malha de triângulos já importada.

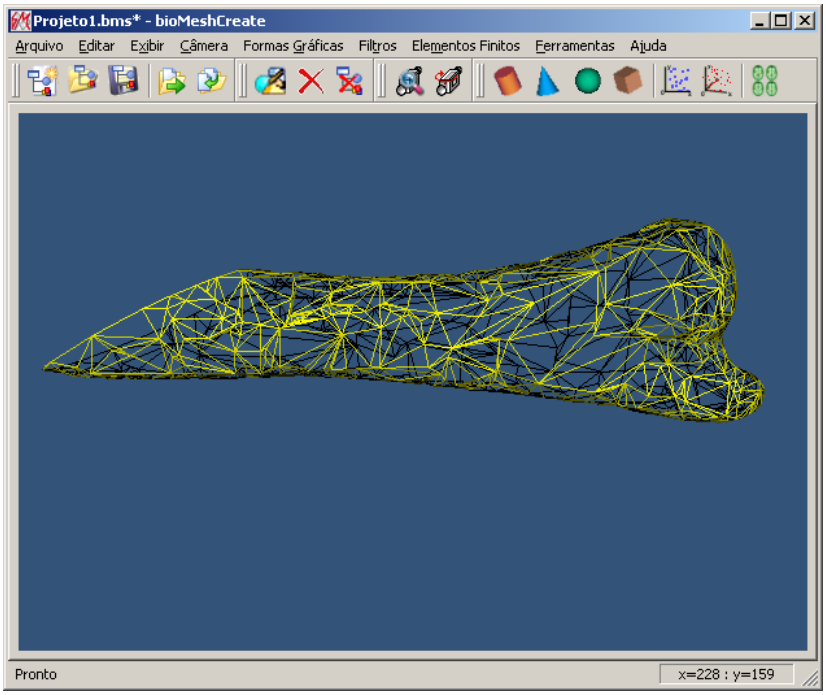


Figura 6.9 – Região de interesse no formato STL importada pelo bioMeshCreate. / Fonte: O autor.

Com o objeto gráfico importado, o próximo passo é aplicar a triangulação de Delaunay 3D sobre ele, a fim de gerar a malha de volume (Figura 6.10). Note que a malha obtida, no entanto, não respeitou os contornos da imagem quando delimitados por regiões côncavas. Na seqüência, os nós e elementos que constituem a malha volumétrica são exportados para um formato compatível com o Ansys, pelo Módulo de Elementos Finitos.

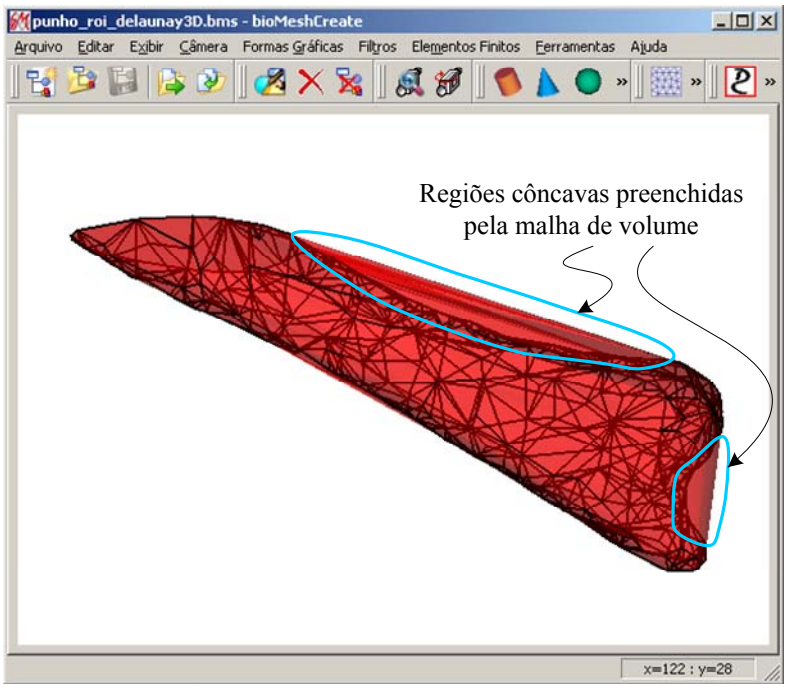


Figura 6.10 – Malha de volume gerada pela triangulação de Delaunay 3D no bioMeshCreate, com 50% de transparência, destacando os contornos da imagem. / Fonte: O autor.

A última etapa consiste no carregamento do modelo gerado, pelo Ansys. Isto é feito através da ferramenta AnsysCtrl. A Figura 6.11 mostra o resultado final desse processo.

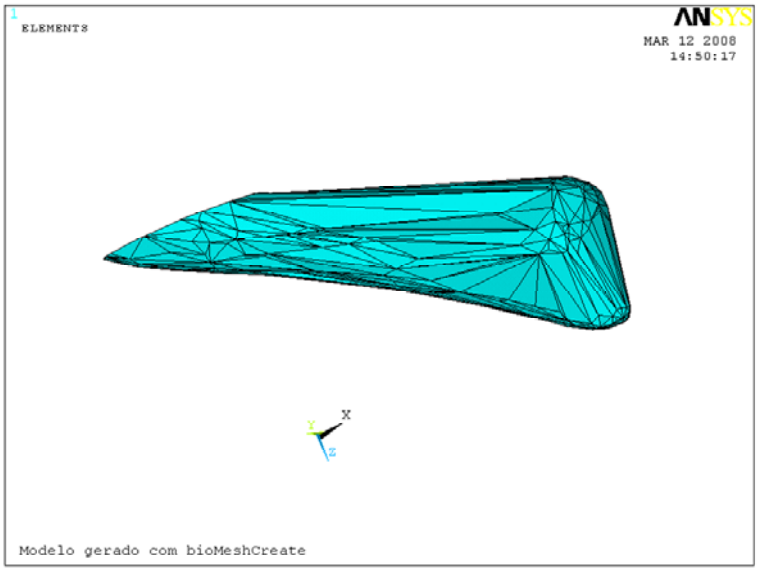


Figura 6.11 – Malha gerada no bioMeshCreate e importada pela ferramenta AnsysCtrl.
Fonte: O autor.

6.4 O PROCESSO DE GERAÇÃO DE MALHA PELO ANSYS

Este processo de geração de malha está baseado na seqüência de passos apresentado pelo fluxograma da Figura 6.3.

Neste caso, a ROI no formato STL foi carregada através de um plug-in do Rhinoceros chamado MeshToSolid (2007), conforme mostrado na Figura 6.12. Este plug-in consegue converter a malha de triângulos em um volume no formato NURBS.

NURBS (Nonuniform Rational B-Splines) é um sistema de modelagem gráfico-computacional que utiliza, além de círculos, linhas e arcos, as splines. É empregado em sistemas CAD para geração de superfícies complexas. São formas matemáticas que permitem a construção desde simples linhas, círculos, arcos até sólidos geométricos de superfícies complexas.

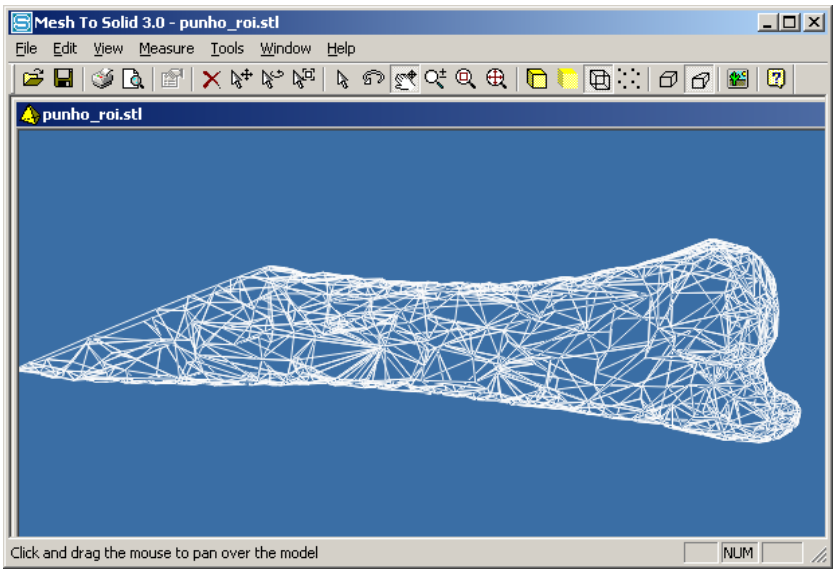


Figura 6.12 – Região de interesse carregada com o plug-in MeshToSolid 3.0, a partir do formato STL.
Fonte: O autor.

Depois disso, o formato NURBS pode ser salvo pelo Rhinoceros como IGES (Initial Graphics Exchange Specification), formato padrão para troca de modelos geométricos entre vários sistemas CAD e CAE, inclusive o Ansys Multiphysics. O formato IGES foi escolhido por ser um padrão aberto e acessível, que embora antigo (iniciado em 1979) ainda é comumente utilizado. Informações adicionais sobre este padrão podem ser encontradas em NIST (2008).

A região de interesse selecionada, já importada pelo Ansys, pode ser observada a seguir (Figura 6.13). Uma vez importada, obteve-se êxito na primeira fase do processo, isto é, a obtenção do modelo geométrico. Seguida por esta, vem a fase do pré-processamento.

No pré-processamento poderá ser necessário realizar alguns ajustes na estrutura, com relação à topologia e à geometria do modelo, pois a transferência de arquivo foi feita com o formato IGES, o qual transfere o modelo apenas como superfície “casca”. Neste trabalho foi tomado como referência o Ansys, mas outros softwares de análise por elementos finitos podem ser utilizados, pois grande parte deles já apresenta módulos para tratamento de modelos importados por outros sistemas.

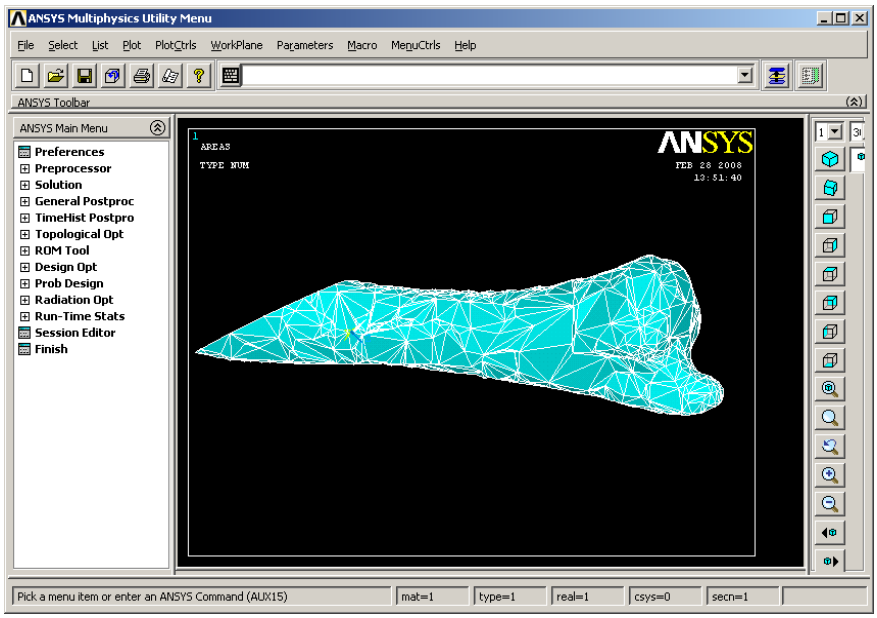


Figura 6.13 – Modelo importado pelo Ansys, a partir do formato IGES.
Fonte: O autor.

No Ansys, a topologia pode ser reparada através de um conjunto de ferramentas agrupadas sob o título de “Topo Repair”. Estas ferramentas reparam pequenos espaços, “gaps”, existentes no modelo. Muitos problemas com o modelo importado podem ser reparados pelas ferramentas de topologia somente.

As ferramentas de reparo à geometria, na documentação do Ansys estão agrupadas sob o título de “CAD Repair” e são em torno de vinte comandos que permitem eliminar a desproporcionalidade entre entidades, tais como pequenas linhas ou áreas. Tais entidades podem causar problemas na geração da malha. Também é possível completar contornos e áreas, agrupá-las e excluí-las, eliminar buracos e cavidades.

A Figura 6.14 mostra a malha gerada sobre a região de interesse importada, num total de 20107 elementos, após vários ajustes na topologia e na geometria do modelo. Estes ajustes incluem a eliminação de ângulos muito agudos presentes em algumas áreas, e pequenos espaços (gaps) não eliminados pelos filtros anteriores. No Ansys, a utilização da APDL, linguagem de scripts para automação, pode facilitar a tarefa. Detalhes sobre a APDL podem ser encontrados em Ansys (2005a).

O pré-processamento ainda prossegue com a definição das propriedades do material, condições de contorno, criação de ósseo-implantes, aplicação de carga e outros mais, pertinentes a esta etapa.

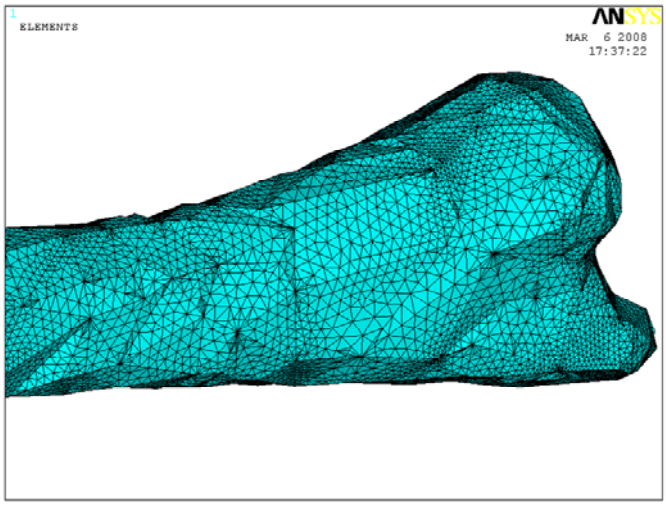


Figura 6.14 – Malha gerada sobre o modelo importado após vários ajustes na topologia e geometria.
Fonte: O autor.

6.5 LIMITAÇÕES E VANTAGENS

O processo de reconstrução 3D pelo bioMesh atingiu bons resultados, embora a criação de novos filtros ainda se faça necessário para melhorar a qualidade da malha de superfície, tal como a eliminação de triângulos muito obtusos e criação de áreas mais distribuídas e homogêneas.

Quanto ao processo de geração da malha de volume através da triangulação de Delaunay 3D, o algoritmo funciona bem para regiões convexas, mas para regiões côncavas ainda tem apresentado problemas por não respeitar os limites da figura e pela qualidade da malha gerada, muito irregular. Refinamentos deverão ser implementados neste algoritmo para corrigir o problema, como o de Alliez et at. (2005). Para mais detalhes, rever seção 3.2.

A segunda técnica, isto é, a geração da malha pelo próprio Ansys, obteve melhores resultados. Embora um pouco mais trabalhoso esse tipo de solução tem como vantagem a portabilidade da análise, pois se aplica a qualquer sistema de análise por elementos finitos, uma vez que quase a totalidade deles aceita a importação no formato IGES e possuem módulos de pré-processamento, enquanto que no primeiro processo, os módulos de exportação/integração foram desenvolvidos especificamente para o Ansys.

7 ANÁLISE COMPARATIVA DOS RESULTADOS

A finalidade deste capítulo é comparar as técnicas de obtenção de modelos para análise por elementos finitos e geração de malha apresentadas nesta pesquisa, com outras existentes. Depois disso, uma validação inicial sobre o programa desenvolvido é realizada, através da comparação dos seus resultados com o de outros sistemas. Esta comparação é dada em nível quantitativo, e em relação à malha de superfície gerada.

7.1 COMPARAÇÃO FACE A OUTROS TRABALHOS

Em trabalhos anteriores a este, a geometria do modelo foi criada com o uso de softwares do tipo CAD como o Solid Edge (2007), ou mesmo através da composição de primitivas gráficas disponíveis nos próprios softwares de análise por elementos finitos. A Figura 7.1, extraída do trabalho de Hayasaki (2007), ilustra um caso onde a região óssea foi construída diretamente no Ansys e, se comparada ao modelo exibido na Figura 6.14, ficam evidentes as vantagens de se utilizar um software de reconstrução 3D.

Tais vantagens referem-se não só à riqueza de detalhes presentes na forma geométrica reconstituída (Figura 6.14), como também à facilidade, rapidez e precisão com que se obtêm os modelos a partir de técnicas de reconstrução 3D por uso de um programa computacional. Por exemplo, para a obtenção do modelo tridimensional da tíbia (Figura 7.1) Hayasaki realizou uma série de medições ao longo do seu comprimento, sendo estas realizadas a cada 5 mm, com o uso de um paquímetro.

Por tratar-se de uma forma geométrica relativamente simples, o modelo pode até representar de forma aceitável a estrutura biomecânica real, mas ao tentar aplicar esta mesma técnica de modelagem a modelos com geometria complexa, tal como um crânio ou uma vértebra da coluna humana, dificilmente serão obtidos resultados sequer próximos da estrutura real. Felizmente existem técnicas mais eficazes para a obtenção de modelos ósseos hoje em dia, através do processamento de imagens médicas digitais, tal como proposto nesta pesquisa.

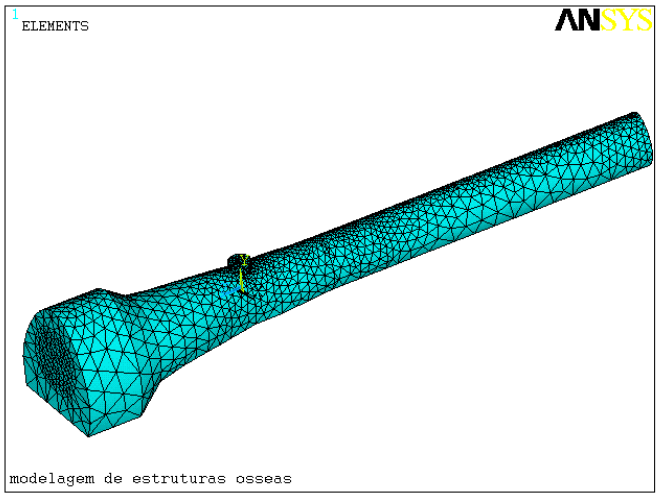


Figura 7.1 – Modelo ósseo da tíbia de coelho construído sem o uso de ferramenta de reconstrução 3D.
Fonte: Hayasaki (2007).

7.2 COMPARAÇÃO FACE A OUTROS SISTEMAS

Um dos passos necessários para tornar o programa confiável é aferir seus resultados. O método adotado aqui foi a comparação dos seus resultados com os de outros sistemas. Atualmente existem vários softwares voltados para visualização científica, reconstrução de imagens e geração de malha. Dentre os sistemas avaliados durante a fase de projeto, dois serão discutidos: o ImageJ (2007) e o 3D-Doctor (2006).

Assim como o desenvolvimento do software, comparar seus resultados, não se mostrou tarefa fácil, pois tanto a reconstrução 3D como a geração de malha são processos abstratos, uma vez que existem vários fatores envolvidos, tanto quantitativos (número de nós e elementos, ângulo de delineamento, fator de relaxação, etc) como qualitativos (distribuição homogênea de áreas, elementos regulares e outros). Nesta comparação, optou-se pelos aspectos quantitativos.

O primeiro teste foi realizado com o ImageJ. Este software é conhecido na comunidade de visualização científica por ser de código fonte aberto e oferecer vários recursos para o tratamento de imagem. Sua implementação foi feita em Java, e um recurso interessante desse software é a sua capacidade de reconhecer plugins (ferramentas auxiliares que executam tarefas específicas) automaticamente, sem a necessidade de instalação, bastando copiá-lo para a pasta plugin, sob o diretório de instalação. Por esse motivo a lista de plugins é bem ampla. Dentre os vários plugins, o que interessou nesta análise foi o

VolumeViewer 1.31.

O VolumeViewer é um plugin especializado em reconstrução de imagens a partir de uma pilha de fatias (stack), que pode estar no formato DICOM. Ele permite controlar a distância, profundidade, fator de escala e filtrar a imagem. Uma aplicação desse plugin pode ser observada na Figura 7.2, abaixo.

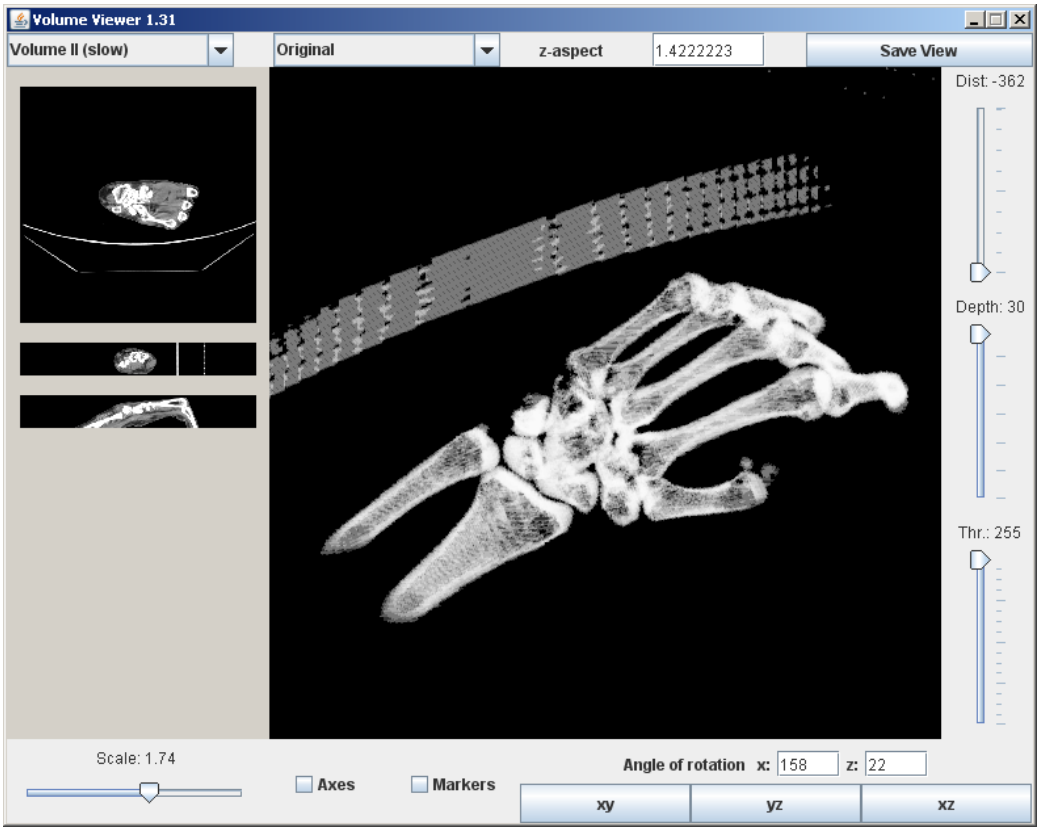


Figura 7.2 – Reconstrução da região do punho com ImageJ v1.38 e o plugin VolumeViewer v1.31.
Fonte: O autor.

O plugin VolumeViewer desempenhou bem a tarefa de reconstrução do volume. Contudo, não foi encontrado no ImageJ uma opção ou plugin capaz de exportar o volume gerado para um formato de dados poligonais ou malha de triângulos. E, muito embora o ImageJ tenha sido usado neste projeto como ferramenta auxiliar em várias situações, principalmente para comparar informações das fatias, como tamanho, profundidade e espaçamento, acabou se mostrando deficiente neste aspecto: exportação da malha poligonal, o que dificultou bastante as comparações.

O segundo teste realizado foi com uma versão de demonstração de um software comercial, o 3D-Doctor. Este software possui recursos avançados, não só de reconstrução 3D, mas também de segmentação das imagens planas. A Figura 7.3 mostra a segmentação automática realizada sobre as 51 fatias que compõem a região do punho humano.

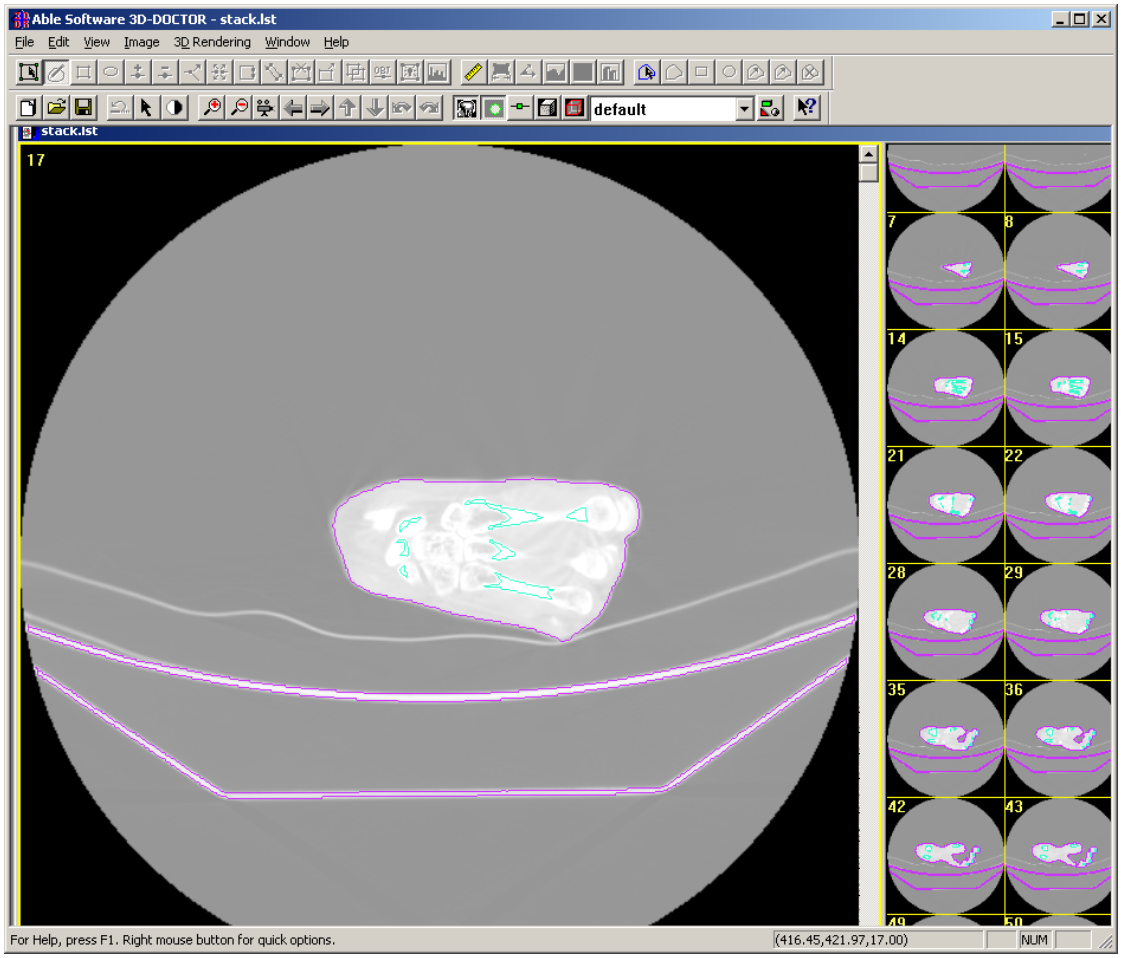


Figura 7.3 – Segmentação das fatias que compõem o punho, executada com 3D-Doctor v3.5.
Fonte: O autor.

Devido ao grande número de recursos que apresenta, desde segmentação, aplicação de filtros de diversas ordens e manipulação de imagens, este software acaba se tornando mais difícil de usar, no entanto, seu processo de reconstrução, se devidamente aplicado, tende a gerar bons resultados, conforme pode ser observado pela Figura 7.4, que representa o mesmo modelo gerado com bioMeshCreate exibido na Figura 6.5 (região óssea), no capítulo anterior. Além disso, o 3D-Doctor permite salvar o modelo como malha de triângulos no formato STL, o que facilitou bastante as comparações, neste caso.

Para chegar aos aspectos quantitativos, os seguintes passos foram dados:

1º) O modelo reconstruído no 3D-Doctor (Figura 7.4) foi salvo no formato STL.

2º) A região de interesse mostrada, sem a aplicação de qualquer filtro, foi selecionada e salva no formato STL com a ajuda do Rhinoceros, devido à facilidade com que ele realiza tais operações, gerando um novo arquivo STL.

3º) O mesmo procedimento foi realizado com o bioMeshCreate, ou seja, o modelo do punho reconstruído, sem a aplicação de quaisquer filtros, foi salvo no formato

STL. Em seguida foi isolada a região de interesse no Rhino, gerando assim um novo arquivo STL com a ROI selecionada.

4º) Um novo projeto foi criado no bioMesh e os dois últimos arquivos STL, contendo a ROI gerada em cada um dos sistemas, foram carregados, conforme pode ser observado na Figura 7.5.

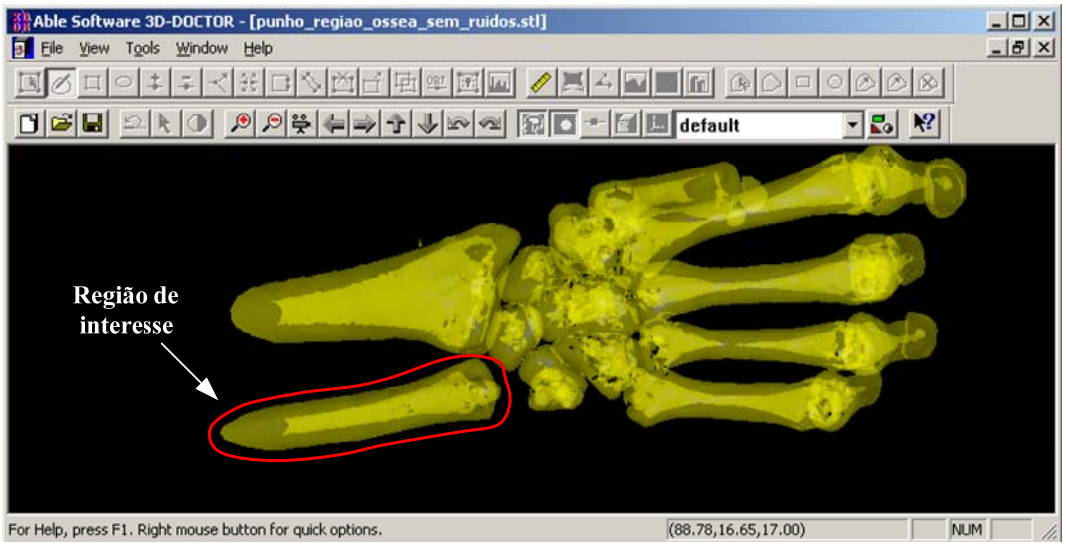


Figura 7.4 – Reconstrução da região do punho com 3D-Doctor.
Fonte: O autor.

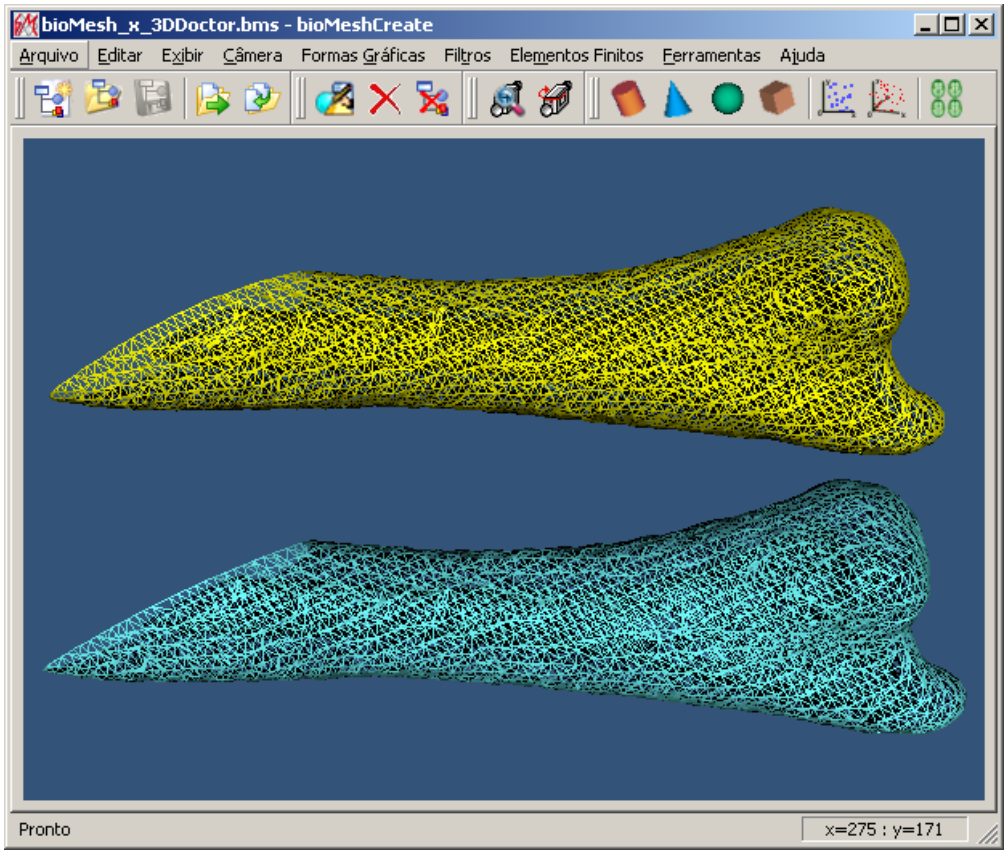


Figura 7.5 – Região de interesse reconstruída no bioMesh (parte superior) e no 3D-Doctor (parte inferior).
Fonte: O autor.

A partir daí foram comparados o número de pontos e células, gerados por ambos os modelos, e o resultado segue conforme a tabela abaixo:

Tabela 3 – Comparação bioMesh x 3D-Doctor.

Programas	Pontos	Células
bioMesh	8.087	16.060
3D-Doctor	7.843	15.577
Diferença (%)	*3,02*	*3,01*

Quanto às dimensões da figura não houve diferenças e quanto às quantidades de pontos e células, a variação registrada foi em torno de 3%, o que pode ser justificado por diversos fatores como os diferentes algoritmos usados em cada sistema de reconstrução, e a variação no valor do filtro de extração de contornos no caso do bioMesh, e segmentação no caso do 3D-Doctor.

Embora os valores obtidos estejam relativamente próximos um do outro, esse tipo de análise, conforme já mencionado, é apenas quantitativa. No futuro, uma comparação em nível qualitativo, entre os modelos obtidos por ambos os sistemas, deverá ser realizada. Tal comparação visa garantir melhor qualidade à malha, com elementos bem distribuídos e homogêneos, e com boa taxa de proporcionalidade.

8 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho procurou integrar-se no universo de tratamento de imagens, levantando os principais métodos numérico-computacionais para reconstrução tridimensional e geração de malhas, encontrando componentes especializados que pudessem auxiliar nessa tarefa, e por fim, elaborando um programa de computador capaz de executá-la: o bioMeshCreate.

O bioMeshCreate, um software destinado à bioengenharia, foi desenvolvido em C++, sob o paradigma da orientação a objeto. O emprego da programação orientada a objetos em sistemas dessa natureza é adequado porque permite escrever programas mais facilmente compreendidos e extensíveis.

O software desenvolvido consegue ler, interpretar e converter imagens obtidas a partir de tomografias computadorizadas ou ressonância magnética em formato médico digital, e também visualizar as fatias bidimensionais. A partir daí são aplicadas técnicas de extração de contorno e isosuperfícies para reconstruir o volume e gerar a malha de triângulos sobre a superfície reconstruída. E uma vez gerada a malha, filtros podem ser aplicados para diminuir a quantidade de triângulos e suavizar os contornos. Depois disso, a malha pode ser exportada para formatos compatíveis com softwares de análise por elementos finitos.

Sua interface gráfica é de uso simples, contando, porém com recursos avançados de visualização científica que permitem modificar facilmente as propriedades de um objeto gráfico, tais como representação, visibilidade, estilo, e até aplicar transformações lineares sobre o mesmo.

Os modelos gerados podem ser exportados para diversos formatos, merecendo destaque o formato de lista de nós e elementos que permite integrar os modelos diretamente com o Ansys. As malhas de triângulos que podem ser exportadas para o formato stereolitography, para uso direto em processos de prototipagem rápida. E também o formato de realidade virtual que exporta toda a cena, gerando arquivos que podem ser distribuídos e visualizados em um browser compatível.

O processo de desenvolvimento foi realizado com o uso de ferramentas de distribuição gratuita, sendo a maioria de código-fonte aberto, o que tornou o processo sem custos no que se refere à aquisição de softwares dessa natureza. Além disso, todas as ferramentas utilizadas são multi-plataforma, e embora a versão inicial tenha sido compilada e testada para a plataforma Windows, versões para as plataformas Unix, Linux e Mac OS

poderão ser disponibilizadas no futuro, se houver necessidade.

A instalação pode ser feita através de um instalador passo-a-passo do tipo wizard, desenvolvido especialmente para a versão Windows 32-bits. Conta também com um manual do usuário abrangendo todas as suas funcionalidades, além de vários projetos de exemplo que foram construídos durante a etapa de simulações e resultados, e que podem ser usados para testar e entender seus recursos.

Na fase final, foram realizadas comparações entre outras pesquisas e entre sistemas já existentes, a fim de apontar as limitações e vantagens do emprego do bioMeshCreate no processo de obtenção dos modelos computacionais com o uso de técnicas de reconstrução 3D, confirmando efetivamente a viabilidade de seu uso.

Contudo, existem algumas limitações, as quais pretende-se eliminar em trabalhos futuros. A principal delas refere-se ao uso de ferramentas de terceiros, ainda necessárias para a fase de seleção da região de interesse e conversão entre alguns formatos de arquivos. A criação de novos filtros também se faz necessário para melhorar a qualidade da malha de superfície, tal como a eliminação de triângulos muito obtusos e criação de áreas mais homogêneas e melhor distribuídas.

Pretende-se também, investir mais no processo de geração da malha de volume, através da implementação de restrições nos algoritmos de triangulação, mais precisamente no algoritmo de triangulação de Delaunay, que ainda tem apresentado problemas com relação a geometrias côncavas e qualidade da malha gerada.

Apesar da necessidade de ajustes e melhorias, os recursos disponíveis nesta versão são de extrema utilidade para a análise por elementos finitos, uma vez que garantem a geração de modelos computacionais com maior fidelidade e precisão, face a outras técnicas apresentadas.

Inserida, portanto, no contexto de Modelagem Científica Computacional, esta pesquisa procurou aplicar a computação a outras áreas do conhecimento, permitindo criar modelos computacionais para situações em que é inviável testar ou medir as diversas soluções possíveis para um fenômeno a partir de modelos experimentais, como por exemplo, tecidos ósseos. Representando assim, para a ciência, uma parcela de contribuição em uma área que continuará, sem sombra de dúvidas, sendo alvo de intensivas investigações.

REFERÊNCIAS

3D-DOCTOR. Vector-Based 3D Medical Modeling and Imaging Software. Able Software Corp. Disponível em: <http://ablesw.com/3d-doctor/ >. Acesso em: 18 dez. 2006.

3D-SLICER. Free open source software package for visualization and image computing. BWH and 3D Slicer. Disponível em: <http://www.slicer.org/>. Acesso em: 08 jul. 2007.

ALLIEZ, Pierre et al. Variational Tetrahedral Meshing. **ACM Trans. Graph,** vol. 24, n. 3, pp. 617-625. July 2005.

ANGLADA, Marc Vigo. An improved incremental algorithm for constructing restricted Delaunay triangulations. **Computers & Graphics**. Vol. 21, n. 2, pp. 215-223, 1997.

ANSYS, Inc. **ANSYS Programmer's Manual**. Disponível em: <http://www.ansys.com/services/ss-documentation-manuals.asp>. Acesso em: 16 set. 2007.

ANSYS, Inc. **APDL Programmer´s Guide**. ANSYS Release 10.0. Canonsburg, PA, 2005.

ANSYS, Inc**. Guide to ANSYS User Programmable Features**. Release 10.0. Canonsburg, PA, 2005.

ANSYS MULTIPHYSICS. General-purpose finite element analysis software. Release 10.0. ANSYS Inc. Disponível em: <http://www.ansys.com/products/multiphysics.asp>. Acesso em: 17 mar. 2007.

ARCHER, Tom; WHITECHAPEL, Andrew. **Visual C++ .NET: Bible**. Indianapolis, Indiana: Wiley Publishing, Inc., 2002.

AVS. Advanced Visual Systems Express. Advanced Visual Systems Inc., MA, USA. Disponível em: <http://www.avs.com/software/soft_t/avsxps.html>. Acesso em: 26 jun. 2007.

BAREQUET, G.; SHAPIRO, D.; TAL, A. Multilevel sensitive reconstruction of polyhedral surfaces from parallel slices. **Visual Computer**, vol. 16, n. 2, pp. 116-133, 2000.

BERG, Mark de et al. **Computational Geometry: Algorithms and Applications**. Third Edition. Springer-Verlag, Heidelberg, 2008.

BERN, Marshall; EPPSTEIN, David. Mesh Generation And Optimal Triangulation. **Computing in Euclidean Geometry**, 2nd ed., World Scientific, pp. 47-123, 1995.

BERTI, Guntram. Image-based unstructured 3D mesh generation for medical applications. In Proceedings of **ECCOMAS 2004,** 2004.

BILBREY, Brian. **Tcl/TK Quick Start**. IBM DeveloperWorks, 2001. Disponível em: <http://www.ibm.com/developerworks/edu/l-dw-linuxtcl-i.html >. Acesso em: 30 jun. 2007.

BLANCHETTE, Jasmin; SUMMERFIELD, Mark. **C++ GUI Programming with Qt 4**. United States: Prentice Hall, 2006.

BOWYER, A. Computing Dirichlet Tessellations. **The Computer Journal**, Heyden & Sons Ltd, vol. 24, n. 2, pp. 162-166, 1981.

BRODLIE, Ken. Scientific Visualization - past, present and future, **Nuclear Instruments and Methods in Physics Research** - A, 354, pp. 104-1 11, 1995.

BROGDEN, Bill; MINNICK, Chris. **Guia do Desenvolvedor JAVA: Desenvolvendo E-Commerce com JAVA, XML e JSP**. Tradução de Mário Moro Fecchio. São Paulo: Pearson Education do Brasil, 2002.

BURIOL, Tiago Martinuzzi. **Processamento e visualização de campos em ambientes virtuais e sistemas CAD 3D aplicados a projetos de iluminação em subestações**. 2006. 123 f. Dissertação (Mestrado em Métodos Numéricos em Engenharia) – Universidade Federal do Paraná. Curitiba, 2006.

BURNS, Marshall. **Automated Fabrication: Improving Productivity in Manufacturing**. Englewood Cliffs: Prentice Hall, 1993.

BUSCHMANN, Frank et al. **Pattern-Oriented Software Architecture: A System of Patterns**. Chichester, UK: Wiley, 1996.

C++ FORUM. **Working Paper For Draft Proposed International Standard For Information Systems - Programming Language C++**. Technical report, American National Standards Institute, 1995.

CAPELLO SOUSA, E. A. **Identificação de Imagens Aplicada a Modelagem de Estruturas Ósseas em Bio-Engenharia**. Relatório Final de Pesquisa - período de 01/01/01 à 31/12/03 - Universidade Estadual Paulista, Faculdade de Engenharia e Tecnologia, Bauru, 2004.

CHAE, Soo-Won; LEE, Gyu-Min. Volume triangulation from planar cross sections. **Computer and Structures**, vol. 72, iss. 1-3, pp. 93-108, Jul. 1999.

CHERNYAEV, E. V. Marching Cubes 33: Construction of Topologically Correct Isosurfaces. **Technical Report CERN CN 95-17**, CERN, 1995. Disponível em: <http://citeseer.ist.psu.edu/chernyaev95marching.html>

CHEW, L. Paul. Guaranteed-Quality Mesh Generation for Curved Surfaces. **Proceedings of the 9th Symposium on Computational Geometry**, ACM Press, pp. 274-280, 1993.

CHIN, Francis Y. L.; WANG, Cao An. Finding the Constrained Delaunay Triangulation and Constrained Voronoi Diagram of a Simple Polygon in Linear Time. **SIAM Journal on Computing,** vol. 28, n. 2, pp. 471-486, 1998.

CLINE, H.E. et al. Two algorithms for the three-dimensional reconstruction of tomograms. **Medical Physics**, Vol. 15, n. 3, pp. 320-327, 1988.

CMAKE. Cross-platform Make. Kitware, Inc. Disponível em: <http://www.cmake.org>. Acesso em: 06 jun. 2007.

DROZDEK, Adam. **Data Structures and Algorithms in C++**. 2 nd. Edition. Pacific Grove, CA: Brooks/Cole, 2000.

DU, Q.; WANG, D. **Recent progress in robust and quality Delaynay mesh generation**. Journal of Computational and Applied Mathematics 195, 2006.

EDELSBRUNNER, H.; MUCKE, E. P. Three-dimensional alpha shapes. **ACM Transactions On Graphics**, 13:43-72, 1994.

EKOULE, A.B.; PEYRIN, F.C.; ODET, C.L. A triangulation algorithm from arbitrary shaped multiple planar contours. **ACM Transactions On Graphics**, vol. 10, n. 2, pp. 182-199, 1991.

EZUST, Alan; EZUST, Paul. **An Introduction to Design Patterns in C++ with QT 4**. Upper Saddle River, NJ: Prentice Hall; Upper Saddle River, NJ: Person Education, Inc., 2007.

FANG, Tsung-Pao; PIEGL, L. A. Delaunay triangulation using a uniform grid. **Computer Graphics and Applications, IEEE**, vol. 13, n. 3, pp. 36-47, May 1993.

FANG, Tsung-Pao; PIEGL, L. A. Delaunay triangulation in three dimensions. **Computer Graphics and Applications, IEEE**, vol. 15, n. 5, pp. 62-69, Sep 1995.

FUCHS, H.; KEDEM, Z. M.; USELTON, S. P. Optimal surface reconstruction from planar contours. **Communications of the ACM**, vol. 20, n. 10, pp. 693-702, 1977.

GANAPATHY, S.; DENNEHY, T. G. A new general triangulation method for planar contours. **Computer Graphics**, vol. 16, n. 3, pp. 69-75, 1982.

GCC, the GNU Compiler Collection. Boston, USA: Free Software Foundation, Inc. Disponível em: <http://gcc.gnu.org/>. Acesso em: 10 dez. 2007.

GEORGE, P.L.; HECHT, F.; SALTEL, E. Automatic 3D mesh generation with prescribed meshed boundaries. **IEEE Transactions on Magnetics**, vol.26, n.2, pp. 771-774, Mar 1990.

GOMES, J.; VELHO, L. **Computação Gráfica - Volume 1**. Série Computação e Matemática, SBM/IMPA, 1998.

HAYASAKI, Cláudio Luís. Modelagem e análise de tensão de estruturas ósseas com **implantes através do Método dos Elementos Finitos**. Dissertação (Mestrado em Engenharia Mecânica) – Universidade Estadual Paulista, Faculdade de Engenharia, Bauru, 2007.

HEGE, H. C. et al. A Generalized Marching Cubes Algorithm Based on Non-Binary Classifications. **Technical Report SC 97-05.** Konrad-Zuse-Zentrum für Informationstechnik Berlin, Dec. 1997.

IBÁÑEZ, L. et al. **The ITK Software Guide**. Second Edition. Updated for ITK version 2.4, November 2005.

IMAGEJ. Image Processiong and Analysis in Java.Version 1.38. Wayne Rasband, Maryland, USA. Disponível em: <http://rsb.info.nih.gov/ij/docs/index.html>. Acesso em: 20 out. 2007.

INVESALIUS. Software livre e público para o tratamento de imagens médicas. CenPRA, MCT. Disponível em: <http://www.softwarepublico.gov.br/>. Acesso em: 02 ago. 2007.

ITK. Insight Toolkit. US National Library of Medicine of the National Institutes of Health. Disponível em: < http://www.itk.org/ >. Acesso em: 06 jun. 2007.

JÓIA FILHO, Paulo. bioMeshCreate – **Manual do Usuário** – Versão 1.0.0, 2008. Disponível em: <http://paulojoia.ddns.com.br>.

JOSUTTIS, Nicolai M. **The C++ Standard Library: A Tutorial and Reference**. Reading, Massachusetts: Addison-Wesley, 1999.

KAGEYAMA, Akira; OHNO, Nobuaki. Tutorial introduction to Virtual Reality: What possibility are offered to our field? Cornell University Library, Kyoto, **Proceedings of ISSS-7**, vol. 2, pp.127-136, March 2005. Disponível em: <http://arxiv.org/abs/physics/0512066>.

KANAGANATHAN, S.; GOLDSTEIN, N.B. Comparison of four-point adding algorithms for Delaunay-type three dimensional mesh generators. **Magnetics, IEEE Transactions on**, vol. 27, n. 3, pp. 3444-3451, May 1991.

KARDOS, I.; HAJDER, L.; CHETVERIKOV, D. Bone Surface Reconstruction From CT/MR Images Using Fast Marching and Level Set Methods. Proc. Joint Hungarian-Austrian **Conference on Image Processing and Pattern Recognition**, Veszprém, pp.41-48, 2005.

KEPPEL, E. Approximating complex surfaces by triangulation of contour lines. **IBM Journal of Research and Development**, vol. 19, pp. 2-11, 1975.

KITWARE, Inc. **The VTK User's Guide: Install, Use and Extend The Visualization Toolkit**, 2006.

KLEMT, A.; INFANTOSI, A. F. C. Método da Superfície na Visualização 3D da Dissecção do Crânio Humano. **Revista Brasileira de Engenharia Biomédica**, vol. 16, n. 1, pp. 21-37, jan/abr 2000.

KOCHAROEN, P. et al. Adaptive mesh generation for mesh-based image coding using node elimination approach. **2005 IEEE International Conference**, vol. 3, n. 16-20 pp. 2052-2056, May 2005.

KOLINGEROVÁ, Ivana; ŽALIK, Borut. Improvements to randomized incremental Delaunay insertion. **Computers & Graphics**, vol. 26, iss.3, pp. 477-490, March 2002.

KOLINGEROVÁ, Ivana; ŽALIK, Borut. Reconstructing domain boundaries within a given set of points, using Delaunay triangulation. **Computers & Geosciences**, vol. 32, iss. 9, pp. 1310-1319, 2006.

KUMAR, Vinod; DUTTA, Debasish. **An assessment of data formats for layered manufacturing.** Advances in Engineering Software. Oxford, UK: Elsevier Science Ltd. v. 28, pp. 151-164, 1997.

LACROUTE, Philippe; LEVOY, Marc. Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation, Proc. **SIGGRAPH '94**, Orlando, Florida, pp. 451-458, July 1994.

LAKARE, Sarang; KAUFMAN, Arie E. OpenVL-The Open Volume Library. **Volume Graphics**, pp. 69-78, 2003

LAM, Roberto; LOKE, Robert; BUF, Hans Du. Alisamento e dizimação de malhas triangulares. **Revista Virtual** - 10º Encontro Português de Computação Gráfica, Lisboa, 2001.

LAU, T. S.; LO, S. H. Finite element mesh generation over analytical curved surfaces. **Computers and Structures**, Elsevier Science Ltd, vol. 59, n. 2, pp. 301-309, 1996.

LEVOY, M. A Hybrid Ray Tracer For Rendering Polygon And Volume Data. **IEEE, Computer Graphics and Applications**, vol. 10, iss. 2, pp. 33-40, march 1990.

LI, Xiang-Yang. Spacing Control and Sliver-free Delaunay Mesh, **Proceedings 9th International Meshing Roundtable**, Sandia National Laboratories, pp. 295-306, October 2000.

LIBERTY, Jessé. **C++ Unleashed**. Indianápolis, U.S.A.: Sams Computer Books, 1998.

LICHTENBELT, Barthold B. A. **Fourier Volume Rendering**. Hewlett Packard Laboratories: United States, 1995.

LIPPMAN, Stanley B. **Essential C++**. Reading, Massachusetts: Addison-Wesley, 1999.

LORENSEN, William E.; CLINE, Harvey E. **Marching Cubes: A high resolution 3D surface construction algorithm**. Computer Graphics, vol. 21, n. 4, pp. 163-169, 1987.

MAGALHÃES, G. M.; PASSARO, A.; ABE, N. M. Geração de Malha de Delaunay Orientada a Objetos. **Anais do Worcomp'2000** – Workshop de Computação, São José dos Campos, SP, 17-18/10, 2000.

MANSSOUR, Isabel Harb; COHEN, Marcelo. Introdução à Computação Gráfica. Porto Alegre: **Revista de Informática Teórica e Aplicada**, vol. 13, n. 2, pp. 43-67, 2006.

MÄNTYLA, M. **An Introduction to Solid Modeling**. Computer Science Press, 1988.

MARTIN, Ken; HOFFMAN, Bill. **Mastering CMake: A Cross-Platform Build System**. 2.2 Edition. Kitware Inc., 2006.

MATLAB. MATrix LABoratory. The MathWorks, Inc.: Massachusetts. Disponível em: <http://www.mathworks.com/products/matlab/>. Acesso em: 05 jun. 2007.

MESHTOSOLID. Software To Convert a Mesh Into a Solid. Version 3.0. Sycode: Goa, India. Disponível em: <http://www.sycode.com/products/index.htm>. Acesso em: 11 out. 2007.

MEURER, Maria Inês et al. Aquisição e manipulação de imagens por tomografia computadorizada da região maxilofacial visando à obtenção de protótipos biomédicos. Radiol Bras, vol. 41, n.1, pp. 49-54, 2008.

MICROSOFT. **Site do Visual C++ 8.0 2005 Express Edition**. Disponível em: <http://www.microsoft.com/express/vc/Default.aspx>. Acesso em: 12 mar. 2007.

MILNE, Philip; NICOLLS, Fred; JAGER, Gerhard de. Visual Hull Surface Estimation, **PRASA2004**, pp. 13-18, Grabouw, Cape Town, 2004.

MILOVANOVIĆ, Jelena; TRAJANOVIĆ, Miroslav. Medical Applications Of Rapid Prototyping. **Facta Universitatis**, Series: Mechanical Engineering, v. 5, n. 1, pp. 79-85, 2007.

MITK. Medical Imaging ToolKit. Disponível em: <http://www.mitk.net/>. Acesso em: 29 jun. 2007.

MORENO, Ignácio Berzal. **Desarrollo de algoritmos de procesamiento de imágenes con VTK**. Grupo de Visión Artificial. Escuela Universitária de Ingeniería Industrial de Madrid, Universidad Politécnica de Madrid, 2004.

MULLER, John Paul. **Visual C++ .NET Developer's Guide**. Berkeley, Califórnia, U.S.A.: McGraw-Hill/Osborne, 2002.

MULLER ADAIME, Leonardo. **Aplicação do Visualization Toolkit para o pós-processamento de análises pelo Método dos Elementos Finitos**. 2005. 126 f. Dissertação (Mestrado em Métodos Numéricos em Engenharia) – Universidade Federal do Paraná. Curitiba, 2005.

NIELSON, Gregory M.; SUNG, Junwon. Interval volume tetrahedrization. Proceedings: **IEEE Visualization '97**: Los Alamitos, CA, USA, pp. 221-228, Oct 1997.

NIST. National Institute of Standards and Tecnology. IGES - The Initial Graphics Exchange Specification. Disponível em: <http://ts.nist.gov/standards/iges/>. Acesso em: 30 jun. 2008.

OPENDX. The Open Source Software Project Based on IBM's Visualization Data Explorer. Visualization and Imagery Solution, Inc. Disponível em: <http://www.opendx.org>. Acesso em: 26 jun. 2007

OPENVL. Library for handling and processing volumetric datasets. Sarang Lakare. Disponível em: <http://openvl.sourceforge.net/>. Acesso em: 01 set. 2007.

OUSTERHOUT, John K. **Tcl and the Tk Toolkit**. Addison Wesley, 1994.

PAGE, Clive G. **Professional Programmer's Guide to Fortran77**. University of Leicester, UK, 2005. Disponível em: <http://www.star.le.ac.uk/~cgp/prof77.pdf>. Acesso em: 16 set. 2007.

PAPADEMETRIS, Xenophon. **An Introduction to Programming for Medical Image Analysis with The Visualization Toolkit**. BioImage Suíte, 2006. Disponível em: <www.bioimage suite.org/vtkbook/xpvtkbook.pdf >. Acesso em: 18 set. 2007.

PARAVIEW. Parallel Visualization Application. Kitware, Sandia National Labs and CSimSoft. Disponível em: <http://www.paraview.org/>. Acesso em: 20 jul. 2007.

PEIXOTO, Adelailson; GATTASS, Marcelo. Reconstrução de Superfícies a partir de Seções Bidimensionais. **Technical Report** MCC 28/00, PUC-Rio, Julho 2000.

POH, C. L.; KITNEY, R. I. Viewing Interfaces for Segmentation and Measurement Results. Proceedings of the 2005 **IEEE, Engineering in Medicine and Biology** 27th Annal Conference, 2005.

QT. **Cross-Platform Rich Client Development Framework**. Oslo, Norway: Trolltech. Disponível em: <http://trolltech.com>. Acesso em: 05 ago. 2007.

RHINOCEROS. NURBS Modeling for Windows. Version 4.0. Robert McNeel & Associates, Seattle, WA,USA. Disponível em: <http://www.rhino3d.com/>. Acesso em: 03 nov. 2007.

RUMBAUGH, James et al. **Modelagem e Projetos Baseados em Objetos**. Tradução de Dalton Conde de Alencar. Rio de Janeiro: Editora Campus, 1994.

RUPPERT, Jim. A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation. **Journal of Algorithms**, vol. 18, n. 3, pp. 548-585, May 1995.

SCHROEDER, W. J.; AVILA, L. S.; HOFFMAN, W. **Visualizing with VTK: A Tutorial**. Kitware, Inc., 2000.

SCHROEDER, W. J.; GEVECI, B.; MALATERRE, M. Compatible Triangulations of Spatial Decompositions. **15th IEEE Visualization**, 2004.

SCHROEDER,Will; MARTIN, Ken; LORENSEN, Bill. **The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics**. 4. ed., Kitware Inc., 2006.

SETHIAN, J. A. **Fast Marching Methods**. University of California, Bekerley, 1998.

SHIRLEY, Peter; TUCHMAN, Allan. A polygonal approximation to direct scalar volume rendering. In Workshop on Volume Visualization, **Computer Graphics**, vol. 24, n. 5, pp. 63-70, San Diego, CA, December 1990.

SILVERIO, CJ; GRAVES, David; HOGUE, Chris. **Fortran 77 Programmer's Guide**. Document Number 007-0711-060. Silicon Graphics, Inc., 1992-1994. Disponível em: <http://www.cepba.upc.es/docs/sgi_doc/SGI_Developer/books/F77_PG/sgi_html/index.html> Acesso em: 16 set. 2007.

SOLID EDGE. Software CAD 3D. V20. Siemens PLM Software: Huntsville, AL, USA. Disponível em: <http://www.solidedge.com>. Acesso em: 25 out. 2007.

SOUZA, Mauren Abreu de; CENTENO, Tania Mezzadri; PEDRINI, Hélio. **Integrando Reconstrução 3D de Imagens Tomográficas e Prototipagem Rápida para a Fabricação de Modelos Médicos**. Revista Brasileira de Engenharia Biomédica, v. 19, n. 2, p. 103-115, 2003.

SOUZA, Mauren Abreu de et al. **Reconstrução de imagens tomográficas aplicada à fabricação de próteses por Prototipagem Rápida usando técnicas de triangulação**. In: Anais do II Congresso Latino Americano de Engenharia Biomédica, 2001, vol. 1., p. 1-4, Havana, 2001.

STROUSTRUP, Bjarne. **The C++ Programming Language**. 3rd. Edition, Reading, Massachusetts: Addison-Wesley, 1997.

SU, Peter; DRYSDALE, Robert L. Scot. A comparison of sequential Delaunay triangulation algorithms. **Symposium on Computational Geometry**, pp. 61-70, 1995.

THE MATHWORKS, Inc. Image Processing Toolbox 6: Perform image processing, analysis, and algorithm development. Disponível em: <http://www.mathworks.com/products/image/>. Acesso em: 14 abr. 2008.

THELIN, Johan. **Foundations of QT Development.** New York, USA: Apress, 2007.

TREECE, G. M.; PRAGER, R. W.; GEE, A. H. Regularised marching tetrahedra: improved iso-surface extraction. **Technical Report CUED/F-INFENG/TR 333**, Cambridge University Engineering Dept, September 1998.

TUY, H.K.; TUY, L.T. Direct 2D Display of 3D Objects, **IEEE Computer Graphics and Applications**, vol. 4, n. 10, pp. 29-33, 1984.

UDUPA, J. K.; ODHNER, D. Shell Rendering, **IEEE Computer Graphics and Applications**, vol. 13, n. 6, pp. 58-67, Nov 1993.

VELHO, L.; GOMES, J. **Sistemas Gráficos 3d**. São Paulo: IMPA, 2001.

VOLVIS. Volume Visualization System. Visualization Lab. Disponível em: <http://www.cs.sunysb.edu/~vislab/volvis_home.html>. Acesso em: 26 jun. 2007.

VRML. Virtual Reality Modeling. Disponível em: <http://www.w3.org/MarkUp/VRML>. Acesso em: 30 jun. 2008.

VTK. The Visualization Toolkit. Kitware Inc. Disponível em: <http://www.vtk.org>. Acesso em: 06 jun. 2007.

VTK DOCUMENTATION. **VTK-Related Papers and Technical Documents.** Kitware Inc. Disponível em: <http://www.vtk.org/documents.php>. Acesso em: 07 jun. 2007.

WAHLIN, Dan. **XML e ASP.NET para Desenvolvedores**. Tradução de Lavio Pareschi. São Paulo: Pearson Education do Brasil, 2003.

WATSON, D. F. Computing the n-Dimentional Delaunay Tessellation with Application to Voronoi Polytopes. **The Computer Journal**. Vol. 24, n. 2, pp. 167-172, 1981.

WELCH, Brent B. **Practical Programming in Tcl and Tk**. 3rd Edition. Prentice Hall, 1994.

WIKIPÉDIA. **Enciclopédia livre**. Disponível em: <http://pt.wikipedia.org>. Acesso em: 15 dez. 2007.

WOLF, Ivo et al. The medical imaging interaction toolkit (MITK): a toolkit facilitating the creation of interactive software by extending VTK and ITK. San Diego, CA, USA: **SPIE 2004**, Volume 5367, pp. 16-27, 2004.

ZHANG, Yongjie; BAJAJ, Chandrajit L.; SOHN, Bong-Soo. Adaptive and quality 3D meshing from imaging data. **Symposium on Solid Modeling and Applications 2003**: Seattle, Washington, USA, pp. 286-291, 2003.

APÊNDICE A – Instalação do BIOMESHCREATE e Conteúdo do CD-ROM

A instalação do programa bioMeshCreate é feita de forma bastante intuitiva através do assistente (wizard). Existem duas versões de instalador, uma para a plataforma Windows XP/2003 e outra para a plataforma 2000 Professional/Server. Os documentos contidos no CD de instalação estão discriminados abaixo.

O CD-Rom de instalação contém os seguintes documentos:

1) Arquivo *bmc-1.0.0-win32_2k.exe*

Este arquivo permite instalar o bioMesh sob as plataformas Windows 2000 Professional e Windows 2000 Server.

2) Arquivo *bmc-1.0.0-win32_xp.exe*

Este arquivo destina-se à instalação no Windows XP e no Windows 2003 Server 32-bits.

3) Arquivo *bmc-1.0.0-manual.pdf*

Manual do usuário do programa bioMeshCreate, contendo todos os recursos existentes no aplicativo, seu funcionamento, possíveis configurações, ferramentas auxiliares, formas gráficas, filtros, módulo de elementos finitos e persistência de dados.

4) Pasta *free_tools*

Esta pasta contém alguns programas de distribuição gratuita utilizados durante a fase de pesquisa / desenvolvimento, entre eles:

- AdbeRdr708_pt_BR.exe → programa Adobe Reader em português para leitura de arquivos no formato pdf.
- CortonaVrmlClient v5.1 → programa para visualização de modelos de realidade virtual (VRML 2.0) integrado ao browser. Foi testado com o Internet Explorer 6 e funcionou corretamente.
- ImageJ v1.38 → ferramenta para tratamento de diversos tipos de imagem.

5) Pasta *bioMeshCreate_data*

Nesta pasta encontram-se arquivos de exemplo que podem ser utilizados para testar o programa bioMeshCreate, entre eles figuram algumas seqüências de arquivos de imagem médica (dcmFiles e rawFiles), geometrias e projetos no formato próprio do bioMesh mostrados na dissertação (bmcModels), alguns modelos gerados pela ferramenta de integração com o Ansys (ansysModels), várias imagens no formato PNG exportadas com bioMeshCreate, arquivos no formato STL (malha de triângulos) e WRL (realidade virtual).

Notas de Instalação:

a) Quando o programa bioMeshCreate é executado pela primeira vez, ele cria a pasta bioMeshCreate_data sob a pasta 'Meus Documentos' (este caminho pode ser alterado através do formulário de opções). Para facilitar a utilização dos arquivos de exemplo, substitua a pasta bioMeshCreate_data criada em 'Meus Documentos' pela que está no CD-Rom.

b) O módulo de reconstrução de volumes permite salvar/recuperar os parâmetros do formulário através dos arquivos PVL. Quando carregar um arquivo PVL de exemplo, na maioria das vezes, é necessário ajustar o caminho dos arquivos de 16-bits (origem) e se preferir, salvar novamente. Para maiores detalhes consulte Manual do Usuário – seção 6.7: Região 1 – persistência dos parâmetros.

c) Foram realizados testes de instalação com sucesso nos seguintes sistemas operacionais:
   - Windows 2003 Server 32-bits (SP2)
   - Windows XP Professional (SP2)
   - Windows 2000 Professional (SP4)
   - Windows 2000 Server (SP4)

d) Esta versão não executa em sistemas operacionais de 16-bits como o Windows 98, nem em sistemas 64-bits como o Windows 2003 Server 64-bits ou Windows Vista.

APÊNDICE B – Scripts APDL para Integração Ansys x bioMeshCreate

Macro-comando: *mcTlbar.mac*

Finalidade: inserir botão de comando na barra de ferramentas do Ansys.

```
1   /NOPR
2   *ABB,BIO_MESH,MCSELMOD
3   /GO
```

Macro-comando: *mcSelMod.mac*

Finalidade: carregar a ferramenta AnsysCtrl e possibilitar a seleção de um modelo.

```
1   /NOPR
2   /syp, C:\Arquiv~1\bioMes~1\ansysCtrl.exe, 'enter_ansys_mode', 'true'
3   fParams =
4
5   *dim,fParams,array,7
6   *vread,fParams(1),'mcArgs','lis',,,1,,,2
7   (7F6.0)
8
9   *if,fParams(1),gt,0,then
10
11    !*** Clear Screen ***!
12    parsav, all, mcSelMod, tmp,
13    /clear, start
14    parres, new, mcSelMod, tmp,
15
16    !*** Preprocessor ***!
17    mcPrep7,fParams(1),fParams(2),fParams(3),fParams(4),fParams(5),fParams(6),fParams(7)
18
19    !*** Replace File ***!
20    replaceE = fParams(5)
21    *if,replaceE,gt,0,then
22      /syp, C:\Arquiv~1\bioMes~1\ansysCtrl.exe, 'mcElems2.lis'
23    *endif
24
25  *endif
26  /GOPR
```

Macro-comando: *mcPrep7.mac*

Finalidade: executar o pré-processamento no Ansys.

```
1                 /NOPR
2                 /TITLE, Modelo gerado com bioMeshCreate
3                 /PREP7
4
5                 !*** Environment ***!
6                 /NERR,0,0,,OFF,0
7                 SHPP, SILENT, ON
8                 *ABSET,,BOTH
9
```

```
!---------------------------------
!           Arguments
!---------------------------------
totNodes = arg1
totElems = arg2
creaE    = arg3
filterE  = arg4
replaceE = arg5
creaA    = arg6
creaV    = arg7

!---------------------------------
!              Nodes
!---------------------------------
nrrang,1,totNodes,1,
nread,'mcNodes','lis'

!---------------------------------
!            Elements
!---------------------------------
*if,creaE,gt,0,then

        !*** Element type ***!
  ET,1,MESH200,8
  myparam = 0

        !*** Without filter ***!
        *if,filterE,eq,0,then
        nrrang,1,totElems,1,
                eread,'mcElems','lis'

        !*** Filter ***!
  *else
          progrBar = nint(totElems/20)
          i = 0
          j = 0

          *do,i,1,totElems,1

            *if,myparam,gt,0,exit
            *if,mod(i,progrBar),eq,0,then
              j = j + 1
              *abcheck,j*5,'Refinando a malha... aguarde'
              *if,_return,gt,0,then
                myparam = 1
              *endif
            *endif

            ERRANG,i,i,1,
            EREAD,'mcElems','lis'

          *enddo
  *endif
  *get,nElems,elem,0,num,max

  !*** New EFile to Replace ***!
  *if,replaceE,gt,0,then
                ewrite, 'mcElems2', 'lis', , 0
        *endif

*endif

!---------------------------------
!             Keypoints
!---------------------------------
*if,(creaA + creaV),gt,0,then

        progrBar = nint(totNodes/20)
        i = 0
        j = 0

        *do,i,1,totNodes

          *if,myparam,gt,0,exit
          *if,mod(i,progrBar),eq,0,then
            j = j + 1
            *abcheck,j*5,'Gerando keypoints... aguarde'
```

```
                          *if,_return,gt,0,then
                            myparam = 1
                          *endif
                        *endif

                        KNODE, 0, i

                      *enddo

            *endif

            !---------------------------------
            !           Areas (only)
    !---------------------------------
    *if,(creaA + creaV),eq,1,then

            progrBar = nint(nElems/20)
            i = 0
            j = 0

            *do,i,1,nElems

              *if,myparam,gt,0,exit
              *if,mod(i,progrBar),eq,0,then
                j = j + 1
                *abcheck,j*5,'Gerando áreas... aguarde'
                *if,_return,gt,0,then
                  myparam = 1
                *endif
              *endif

              N1 = NELEM(i,1)
              N2 = NELEM(i,2)
              N3 = NELEM(i,3)
              N4 = NELEM(i,4)
              A,N1,N2,N3
              A,N1,N2,N4
              A,N1,N3,N4
              A,N2,N3,N4

            *enddo
            /color,area,oran
            *get, nAreas, area, 0, count

    *endif

    !---------------------------------
    !         Areas/Volumes
    !---------------------------------
    *if,(creaA + creaV),eq,2,then

            progrBar = nint(nElems/20)
            i = 0
            j = 0

            *do,i,1,nElems

              *if,myparam,gt,0,exit
              *if,mod(i,progrBar),eq,0,then
                j = j + 1
                *abcheck,j*5,'Gerando áreas/volumes...aguarde'
                *if,_return,gt,0,then
                  myparam = 1
                *endif
              *endif

              N1 = NELEM(i,1)
              N2 = NELEM(i,2)
              N3 = NELEM(i,3)
              N4 = NELEM(i,4)
              V,N1,N2,N3,N4,0,0,0,0

            *enddo
            /color,area,oran
            *get, nAreas, area, 0, count
            /color,volu,gree
            *get, nVols, volu, 0, count
```

```
164
165     *endif
166
167     !*** Environment ***!
168     *ABFINISH
169     /NERR,DEFA
170     /UIS, MSGPOP, 0
171
172     !----------------------------------
173     !              Results
174     !----------------------------------
175     *if,myparam,gt,0,then
176
177       *msg,warn
178         Processo de importação abortado... Solução incompleta.
179
180     *else
181
182       !*** Without Filter ***!
183       *if,filterE,eq,0,and,nElems,ne,totElems,then
184                       *msg, error, totElems, nElems
185                  Falha ao importar elementos.%/&
186                  %/&
187                  Elementos lidos: %I %/&
188                  Elementos importados: %I %/&
189                  %/&
190                  Tente a opção filtrar elementos da próxima vez,%/&
191                  pois ela garante o refinamento da malha pelo Ansys.
192
193       !*** Filter ***!
194       *else
195                *msg, note, totNodes, totElems, totElems-nElems, nElems, nAreas, nVols
196                  Processo de importação de modelo concluído. %/&
197                  %/&
198                  Nós importados: %I %/&
199                  Elementos lidos: %I %/&
200                  Elementos filtrados: %I %/&
201                  Total de elementos importados: %I %/&
202                  Total de áreas construídas: %I %/&
203                  Total de volumes definidos: %I
204
205                /VIEW,1,1,1,1
206                /ANGLE,1
207                /REPLOT,FAST
208
209                *if,creaE,gt,0,then
210                  eplot
211                *endif
212              *endif
213
214     *endif
215
216     FINISH
217     /GOPR
```

SÚMULA DE TRABALHOS DESENVOLVIDOS

- **Trabalho Aceito em Congresso - com arbitragem**

JÓIA FILHO, Paulo; CAPELLO SOUSA, Edson A. **Reconstrução e Geração de Malha em Estruturas Biomecânicas Tridimensionais para Pré-Processamento e Análise por Elementos Finitos**. V CONEM, 25 a 28/08/08, Salvador Bahia, 2008.

- **Trabalho Submetido à Revista - com arbitragem**

JÓIA FILHO, Paulo; CAPELLO SOUSA, Edson A. **Reconstrução e Geração de Malhas em Estruturas Biomecânicas Tridimensionais para Análise por Elementos Finitos**. Revista Brasileira de Engenharia Biomédica – RBEB, 2008.

- **Programas Computacionais Desenvolvidos**

bioMeshCreate, v.1.0.0, Programa para reconstrução 3D e geração de malhas, disponível em: <paulojoia.ddns.com.br>.

DicomConverter, v.1.0.0, Ferramenta para conversão de arquivos de imagem médica digital, disponível em: <paulojoia.ddns.com.br>.

AnsysControl, v.1.0.0, Módulo de integração de modelos bioMeshCreate x Ansys Multiphysics, disponível em: <paulojoia.ddns.com.br>.

- **Outros Trabalhos**

JÓIA FILHO, Paulo. bioMeshCreate – Manual do Usuário – Versão 1.0.0, 2008. Disponível em: <http://paulojoia.ddns.com.br>.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo