Escola Politécnica da Universidade de São Paulo

Departamento de Engenharia Mecatrônica

PMR 2550 – Projeto de Formatura II

"Construção de Modelo e Simulação de CAE Aplicados a Próteses Odontológicas"

Orientador: Prof. Dr. Emílio Carlos Nelli Silva

Aluna: Heloísa Alves Fonseca

São Paulo

2004

Nota final
7.7 (sete e sete)

Escola Politécnica da Universidade de São Paulo

Departamento de Engenharia Mecatrônica

PMR 2550 – Projeto de Formatura II

"Construção de Modelo e Simulação de CAE Aplicados a Próteses Odontológicas"

Orientador: Prof. Dr. Emílio Carlos Nelli Silva

Aluna: Heloísa Alves Fonseca

São Paulo

2004

À minha mãe, Maisa, por acreditar em mim.

AGRADECIMENTOS

Ao meu orientador, Professor Doutor Emílio Carlos Nelli Silva, pelo seu empenho e dedicação dados durante o desenvolvimento deste trabalho.

A Fernando Jaruche Nunes, também aluno da Escola Politécnica, por todo seu apoio e colaboração.

Ao Departamento de Engenharia Mecatrônica da Escola Politécnica da Universidade de São Paulo.

A todos que contribuíram para a execução deste trabalho.

RESUMO

Com o avanço da tecnologia, nos últimos anos vem surgindo um novo campo de estudos interdisciplinar entre a Medicina e a Engenharia. Em geral, os trabalhos nessa área visam à aplicação de técnicas da Engenharia como auxiliares em diagnósticos médicos. Aplicações na área odontológica têm surgido com freqüência, principalmente nas questões relacionadas a implantes odontológicos.

Nesse campo diversos estudos relacionados à aplicação do Método dos Elementos Finitos (MEF) vêm sendo desenvolvidos. O MEF é um método numérico que visa à solução de problemas com infinitas variáveis. Contudo, ainda há problemas que restringem a aplicação do MEF em ramos como a medicina e a odontologia. O principal deles é o tempo de modelagem elevado, incompatível com os tratamentos da área de saúde.

Alguns métodos alternativos têm sido desenvolvidos com o objetivo de minimizar os problemas, viabilizando o uso do MEF. Dentre esses métodos destaca-se o uso do tomógrafo, um aparelho capaz de obter imagens das seções transversais de um corpo sólido. Através dessas imagens é proposto reconstruir um sólido em três dimensões (3-D), ou seja, um modelo digital do que foi fotografado pelo tomógrafo. O resultado é semelhante ao obtido pela modelagem tradicional, no entanto o tempo despendido no processo é consideravelmente menor.

Assim, a proposta deste projeto é o estudo do uso de imagens tomográficas para reconstrução de próteses odontológicas, através de análises mecânicas embasadas no Método dos Elementos Finitos. O objetivo final do trabalho é desenvolver um software que viabilize o uso do MEF por especialistas da área odontológica no tratamento de seus pacientes.

Para tal, à reconstrução de imagens tomográficas devem ser integradas técnicas de Elementos Finitos e de Visualização 3-D; além de uma interface amigável para o usuário. Neste trabalho, a interface deve ser desenvolvida em linguagem C++, já prevendo uma integração com o método de visualização, para o qual é proposto o uso do programa VTK (biblioteca de C++). Além disso, as técnicas de reconstrução e de MEF já existentes devem ser integradas ao software. Como resultado final é esperado um software independente e de fácil uso para os profissionais da área odontológica.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SIGLAS E ABREVIAÇÕES

2-D: elemento ou visão bidimensional, com duas direções fundamentais;

3-D: elemento ou visão tridimensional, com três direções fundamentais;

CAD: Computer Aided Design;

CAE: Computer Aided Engineering;

MEF: Método dos Elementos Finitos;

VTK: Visualization Tool Kit.

1. INTRODUÇÃO

Com o avanço da tecnologia, nos últimos anos vem surgindo um novo campo de estudos interdisciplinar entre a Medicina e a Engenharia. Em geral, os trabalhos nessa área visam à aplicação de técnicas da Engenharia como auxiliares em diagnósticos médicos. Como resultado, a diagnose de algumas doenças pode se tornar mais simples, dando mais chances de cura ao paciente. Além disso, alguns estudos têm o objetivo de dar ao profissional da área médica evidências que o assistam na análise clínica dos pacientes.

Nesse campo diversos estudos relacionados à aplicação do Método dos Elementos Finitos (MEF) vêm sendo desenvolvidos. O MEF é um método numérico que visa à solução de problemas com infinitas variáveis através da divisão do mesmo em problemas discretos com variáveis finitas. Em suma, o espaço contínuo a ser estudado é subdividido, dando origem aos chamados elementos finitos; assim, o equacionamento do problema passa a ser feito para cada um desses elementos.

O MEF tem se destacado como uma ferramenta efetiva na análise do corpo humano; seu uso permite, por exemplo, que antes de realizar a colocação de uma prótese, o clínico possa analisar e minimizar as seqüelas físicas trazidas ao paciente. A Figura 1 ilustra esse tipo de aplicação; nesse caso trata-se da modelagem de uma prótese, em tom esverdeado, para a mandíbula previamente reconstruída.

Figura 1 - Mandíbula reconstruída com prótese

Aplicações na área odontológica têm surgido com freqüência, principalmente nas questões relacionadas a implantes odontológicos. A seleção adequada do modelo do implante, bem como do seu posicionamento, são vitais para sua durabilidade. Um carregamento excessivo ao redor do mesmo pode ocasionar micro fraturas no osso perfurado, perda do implante e eventual impossibilidade de re-implante na região afetada. [1, 2]

Estudos demonstram que é possível calcular a localização e a intensidade das tensões ocorridas na interface entre a mandíbula e o implante [3, 4, 5]. O MEF já foi aplicado para evidenciar que essas tensões dependem de fatores bastante complexos, tais como: a densidade do osso onde os implantes foram incrustados, o grau de deslocamento ocorrido sob carregamento, e a relação que há entre implantes adjacentes. Uma conseqüência conclusiva é que as ocorrências de problemas em implantes não podem ser generalizadas, pois seu resultado depende diretamente de características próprias de cada paciente [3]. Há também pesquisas voltadas à influência do número de implantes e da distribuição dos mesmos ao longo da mandíbula [6]. Nesse caso, alguns modelos de elementos finitos são construídos, variando-se quantidade e posição dos implantes. Isso permite que análises comparativas sejam feitas, embasando a escolha da melhor configuração. Como conseqüência, é esperado um aumento da durabilidade das peças implantadas, além de maior conforto para o paciente. Também se destacam estudos relacionados à influência da inclinação [1], do material utilizado na fabricação[2], e da pré-carga aplicada durante a fixação dos implantes [7].

Outra área da odontologia que apresenta algumas pesquisas envolvendo a aplicação de elementos finitos é a de restauração. Aqui, modelos de MEF permitem a comparação entre diferentes técnicas e materiais [8], porém seu uso é pouco difundido.

Contudo, ainda há problemas que restringem a aplicação do MEF em ramos como a medicina e a odontologia. O principal deles é o tempo de modelagem elevado, incompatível com os tratamentos da área de saúde. Alguns métodos alternativos têm sido desenvolvidos com o objetivo de minimizar os problemas, viabilizando o uso do MEF. Dentre esses métodos destaca-se o uso do tomógrafo, um

aparelho capaz de obter imagens das seções transversais de um corpo sólido, como ilustrado na figura abaixo.

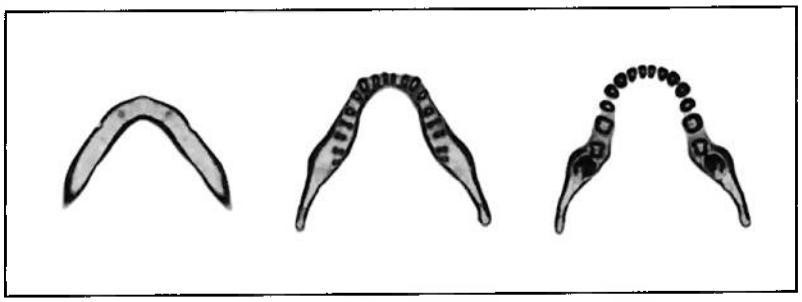


Figura 2 - Exemplos de imagens de uma mandíbula obtidas por tomógrafo

O funcionamento do tomógrafo se dá a partir de uma série de medições consecutivas em várias direções no plano da seção desejada. Tais medições podem ser feitas de muitas maneiras, porém os tipos mais disseminados de tomografia são: por raios-X, por ressonância magnética e por impedância elétrica[10].

A reconstrução do sólido tomografado é feita usando técnicas de reconstrução algébrica aplicadas às imagens obtidas. As seções transversais obtidas do tomógrafo são transformadas em matrizes numéricas, ao que se chama digitalização. A partir das imagens digitalizadas é possível melhorar a sua resolução e também criar novas seções intermediárias através de funções que aproximam as cores dos *pixels* [9]. Esse segundo processo denomina-se discretização. Depois de digitalizadas e discretizadas as imagens podem ser usadas na reconstrução do sólido original, que poderá ser visualizado por um programa comercial de elementos finitos.

Na engenharia existem algumas pesquisas que associam o MEF ao uso dos tomógrafos, como por exemplo, o estudo das interfaces internas de um sólido multi-materiais. Em casos em que não se pode ter acesso direto a essas interfaces, o estudo é feito através da reconstrução do sólido baseada em imagens tomográficas [10]. A figura abaixo, por exemplo, faz parte do estudo de uma múmia Egípcia encontrada na Itália[16]; nesse caso o uso da tomografia associada à reconstrução tridimensional permite que sejam realizados estudos aprofundados sem dano à peça arqueológica.

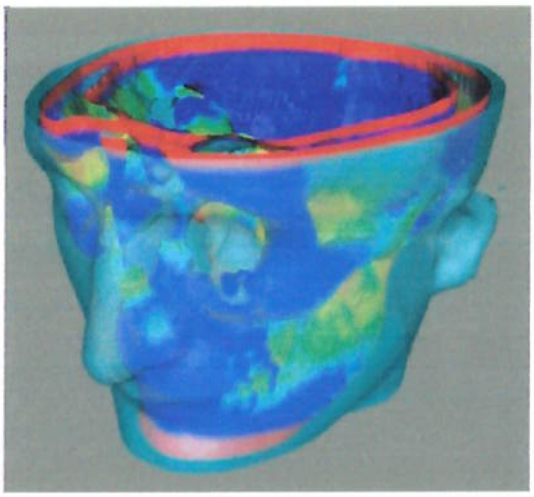


Figura 3 - Reconstrução computacional feita a partir da múmia

Focalizando as aplicações desse método na área médica, um exemplo é o monitoramento de pacientes com osteoporose [11]. Nesse caso são usadas imagens de raios-X para a reconstrução da coluna vertebral do paciente. O Método dos Elementos Finitos permite uma análise tanto da densidade como da geometria dos ossos, viabilizando o prognóstico de fraturas devidas à baixa resistência dos ossos.

Assim, a proposta deste projeto é o desenvolvimento de um software que viabilize o uso do MEF por especialistas da área odontológica no tratamento de seus pacientes. As imagens tomográficas devem ser usadas como ponto de partida para a reconstrução computacional das próteses, que poderão ser visualizadas com o auxílio do VTK.

2. JUSTIFICATIVA E MOTIVAÇÃO

O uso de modelagem computacional na área médica vem crescendo consideravelmente nos últimos anos. Diversos profissionais utilizam softwares especializados em tarefas como cirurgias e colocação de próteses. Em geral, os softwares presentes nessa área baseiam-se no Método dos Elementos Finitos (MEF), o qual permite uma simulação computacional do problema. No ramo odontológico há bastante interesse dos profissionais por esse tipo de software, pois tais ferramentas podem auxiliá-los no estudo das conseqüências que procedimentos, como a colocação de uma prótese ou alterações na arcada dentária, podem trazer ao paciente.

No entanto, o custo para se construir um modelo de elementos finitos aplicado a partes do corpo humano acaba inviabilizando essa técnica. Um dos problemas é que simulações com MEF necessitam de investimentos financeiros, pois são necessários softwares específicos, além de profissionais especializados em modelagem de CAD. O segundo e principal problema é o fator tempo, visto que modelos de qualquer parte do corpo humano são bastante complexos, e por isso demandam um número grande de horas de trabalho para serem reproduzidos computacionalmente. Essa delonga do processo de construção de um modelo é incompatível com o tratamento médico de pacientes, pois não é possível aguardar o tempo de modelagem para se dar um diagnóstico.

O uso de imagens tomográficas tem o propósito de reduzir o problema exposto sobre o tempo, e conseqüentemente viabilizar a técnica de aplicação do MEF à medicina. A tomografia já é um processo consolidado na área médica, sendo frequentemente usado na confirmação de diagnósticos, e, consequentemente, familiar aos profissionais da área. O resultado gerado por um tomógrafo é um conjunto de imagens transversais, tal como apresentado na Figura 4, com espaçamento físico que pode ser determinado de acordo com a análise desejada.

Através dessas imagens é proposto reconstruir um sólido em três dimensões (3D), ou seja, um modelo digital do que foi fotografado pelo tomógrafo. O resultado é semelhante ao obtido pela modelagem tradicional, no entanto o tempo despendido no processo é consideravelmente menor. Com esse modelo 3-D baseado em imagens

tomográficas, será possível aplicar o Método dos Elementos Finitos como uma ferramenta auxiliar das investigações clínicas.

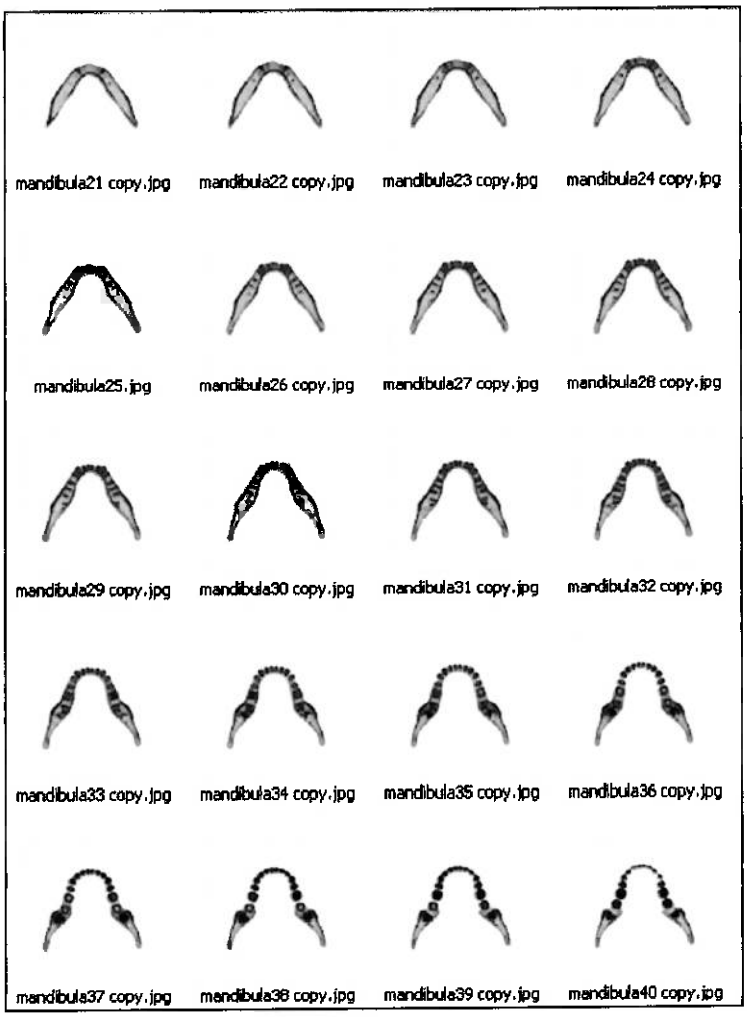


Figura 4 - Conjunto de imagens resultantes da tomografia de uma mandíbula

3. OBJETIVOS

Este projeto tem como objetivo o desenvolvimento de um software capaz de construir e simular um modelo de CAE aplicado a uma prótese dentária.

O modelo deve ser construído a partir das imagens das seções transversais do objeto. O software deve trabalhar com imagens em formato bitmap (*.bmp) de 8 bits, pois as imagens geradas pela maior parte dos tomógrafos seguem esse padrão.

O módulo de visualização é baseado no VTK, para que seja possível uma interação em tempo real com o usuário. Além disso, a integração da implementação de elementos finitos permite que o software seja independente de outros, viabilizando o seu uso pelos profissionais da área odontológica.

4. RECONSTRUÇÃO DE IMAGENS

Diversos tipos de imagens podem ser usados como base da reconstrução de uma forma tridimensional. Tratando do corpo humano, os meios mais difundidos de obtenção dessas imagens são raios-X, ultra-som e tomografia [12], pois o custo desses procedimentos é relativamente baixo. O resultado gerado por qualquer um desses métodos é uma gama de imagens bidimensionais representativas das seções transversais da região trabalhada. Por outro lado, existem métodos bem mais avançados, capazes de obter imagens tridimensionais do corpo humano, tais como a tomografia computadorizada 3-D e a ressonância magnética; no entanto a sua aplicação é restrita devido ao custo elevado e à infra-estrutura requerida.

Para o processo de reconstrução de sólidos se tornar acessível à classe médica é necessária a aplicação de métodos com um custo financeiro não muito elevado. Por essa razão, e pela infra-estrutura disponível, optou-se pelo uso de um tomógrafo no processo de obtenção das seções 2-D. Um exemplo desse tipo de aparelho pode ser visto na Figura 5.

Figura 5 - Exemplo de um tomógrafo

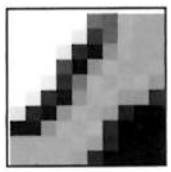
A reconstrução a partir de imagens bidimensionais é feita relacionando-se cada *pixel* das imagens 2-D a uma coordenada 3-D do sólido original, ou seja, devem ser determinadas equações para cada linha tridimensional correspondente a uma união de *pixels* de diferentes imagens. Para um resultado de boa acurácia, é necessário trabalhar com um elevado número de seções transversais, a fim de que detalhes da superfície tomografada não sejam perdidos [12]. Porém, quanto maior o

número de imagens utilizadas, maior o tempo computacional requerido; assim, alguns métodos alternativos que permitam uma redução do número de imagens tornam-se importantes. Neste trabalho é utilizado um programa para recriar as seções transversais existentes entre as imagens digitalizadas.

Resumidamente, após as imagens das seções terem sido obtidas, o processo de reconstrução apresentado neste trabalho segue os seguintes passos:

4.1. Digitalização das imagens

Cada *pixel* é transformado em um valor numérico, entre zero e um, representativo da sua cor. Assim, pode-se dizer que cada seção passa a ser representada por uma matriz numérica, tal como exemplificado na Figura 6.



(a)

0.0	0.0	0.0	0.0	0.0	1.0	0.5	0.5	0.5	0.5
0.0	0.0	0.0	0.0	1.0	1.0	0.5	0.5	0.5	0.5
0.0	0.0	0.0	1.0	1.0	0.5	0.5	0.5	0.5	0.5
0.0	0.0	0.0	1.0	0.5	0.5	0.5	0.5	0.5	0.5
0.0	0.0	1.0	1.0	0.5	0.5	0.5	0.5	0.5	1.0
0.0	1.0	1.0	0.5	0.5	0.5	0.5	1.0	1.0	1.0
1.0	1.0	0.5	0.5	0.5	0.5	1.0	1.0	1.0	1.0
0.5	0.5	0.5	0.5	0.5	0.5	1.0	1.0	1.0	1.0
0.5	0.5	0.5	0.5	0.5	1.0	1.0	1.0	1.0	1.0
0.5	0.5	0.5	0.5	1.0	1.0	1.0	1.0	1.0	1.0

(b)

Figura 6 - Exemplo de imagem digitalizada: a) Imagem BITMAP; b) Matriz resultante da digitalização

4.2. Discretização

Esse processo é responsável por recriar as seções intermediárias perdidas no processo de tomografia. Isso é feito através de funções matemáticas baseadas nas matrizes numéricas descritas anteriormente. Diversos métodos numéricos podem ser utilizados na obtenção dessas funções; mas, neste caso, as funções utilizadas são baseadas no Método de Shepard [9].

4.2.1. Método de Shepard

O princípio básico do método de discretização é aproximar os *pixels* conhecidos por funções, evitando oscilações que alterem a forma inicial do sólido. O Método de Shepard constrói essas funções baseando-se na combinação entre os valores assumidos em cada ponto (c_1, c_2,..., c_n), que na aplicação traduz-se pela cor assumida por cada *pixel*, além de um peso associado a cada um deles. O formato da função é o seguinte:

$$f(x) = c_1\phi_1(x) + c_2\phi_2(x) + ... + c_n\phi_n(x)$$

Onde c_1, c_2,..., c_n são os valores de f(x) nos pontos x_1, x_2,..., x_n; e os termos $\phi_1(x), \phi_2(x), ..., \phi_n(x)$ são as funções de forma da função. Uma função de forma é formada por uma função polinomial com um peso associado $w_i(x)$, quando x_i é o ponto em questão.

$$\phi_i(x) = \left(a_0(x) + x_i a_1(x) + x_i^2 a_2(x) + ... + x_i^n a_n(x)\right) w_i(x), \text{ com i = 1, 2,..., n}$$

Para o caso mais simples, onde n é igual a 1, as funções de forma tornam-se lineares:

$$\phi_i(x) = \left(a_0(x) + x_i a_1(x)\right) w_i(x), \text{ com i = 1, 2,..., n}$$

E, segundo o Método de Shepard [9], suas constantes podem ser obtidas do sistema abaixo.

$$\begin{bmatrix} w_1(x) + w_2(x) + ... + w_n(x) & x_1 w_1(x) + x_2 w_2(x) + ... + x_n w_n(x) \\ x_1 w_1(x) + x_2 w_2(x) + ... + x_n w_n(x) & x_1^2 w_1(x) + x_2^2 w_2(x) + ... + x_n^2 w_n(x) \end{bmatrix} \begin{Bmatrix} a_0(x) \\ a_1(x) \end{Bmatrix} = \begin{Bmatrix} 1 \\ x \end{Bmatrix}$$

Já o peso da função de forma pode variar de zero a um, e é dado por:

$$w_i(x) = e^{-\alpha(x - x_i)}$$

Assim, quando o valor x da função tende a x_i, o peso associado a x tende ao seu valor máximo, ou seja, 1; fazendo com que esse ponto tenha bastante influência na função. No caso contrário, quando x se distancia muito se x_i, seu peso tende a zero, e o valor do ponto passa a ter pouca relevância. O fator α determina a proximidade entre a função e os pontos desejados, portanto, a elevação do valor de α forçará a função a passar cada vez mais próxima dos pontos exatos. Os gráficos abaixo ilustram como esse fator influencia na reconstrução do contorno do objeto.

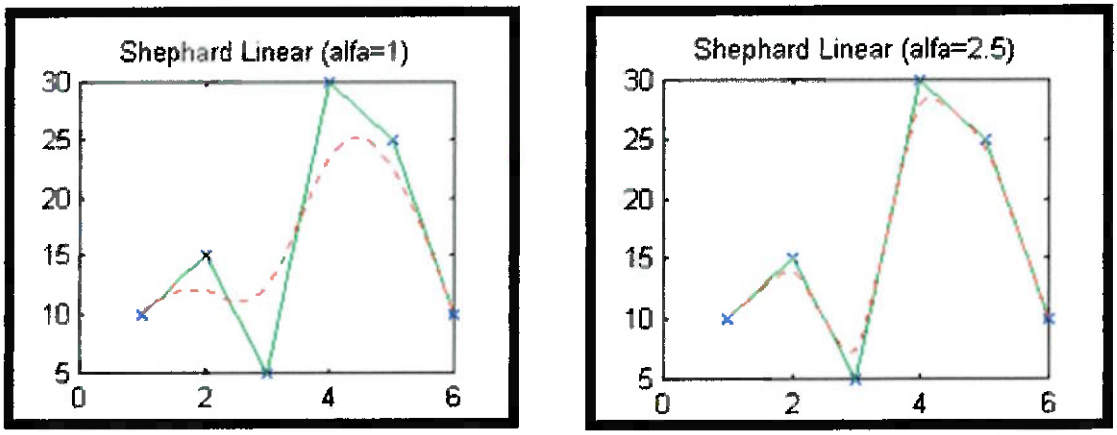


Figura 7 - Funções Aproximadoras do Método de Shepard

Mais detalhes sobre o Método de Shepard e análises comparativas que justificam a sua escolha podem ser encontradas na referência [9].

4.3. Conversão

Por fim, as matrizes numéricas recolhidas e criadas devem ser unidas em um arquivo compatível com os programas de visualização disponíveis. Ao serem unidos, os elementos bidimensionais, *pixels*, passam a representar elementos tridimensionais, denominados *voxels* (vide Figura 8). Esses *voxels* são, então, unidos por um padrão de conectividade, determinado a partir dos nós envolvidos nos elementos conectados.

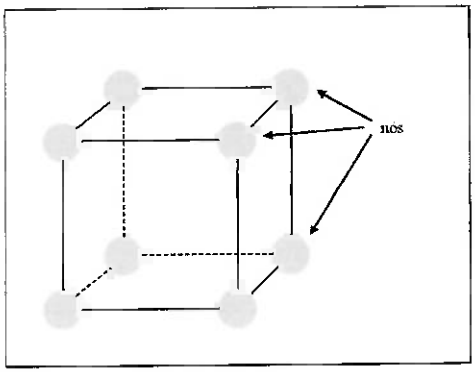


Figura 8 - Esquema de um VOXEL

5. MÉTODO DOS ELEMENTOS FINITOS

O MEF é um método numérico muito útil na resolução de problemas com infinitas variáveis ou problemas de domínio contínuo. A idéia básica desse método é a divisão do problema inicial em diversos subproblemas que podem ser equacionados.

Pode-se aferir que o MEF é baseado em dois conceitos principais: [14] (1) Sua solução baseia-se na resolução de integrais das equações; (2) Essa solução é aproximada por uma função definida em um subdomínio resultante da discretização do domínio original. Ou seja, o método procura uma solução aproximada que reduza ao mínimo o erro na aproximação da equação.

Para se resolver o problema de minimização do erro, a grandeza em questão deve ser aproximada por uma função, por exemplo, um polinômio. A idéia básica dessa solução seria definir um polinômio que compreendesse todo o domínio da equação; no entanto, esse método, chamado Método de Ritz, apresenta a desvantagem de gerar distorções quando há variações bruscas na equação original. Como num problema prático a equação original é uma incógnita, não é possível prever essas distorções. A fim de evitar isso, o domínio do problema é dividido então em subdomínios, cada qual passa a ser aproximado pelo melhor polinômio. Essa divisão do domínio é denominada discretização e os subdomínios resultantes são os elementos finitos propriamente ditos.

O MEF pode ser implementado para uma, duas ou três dimensões; porém, os conceitos envolvidos são basicamente os mesmos. Este trabalho demanda uma aplicação tridimensional do método, e por isso será apresentada uma breve explanação sobre o mesmo. Os elementos envolvidos são cubos isoparamétricos de 8 nós, tal como os *Voxels* descritos anteriormente. Demonstra-se que uma grandeza física do elemento pode ser aproximada por um polinômio tri-linear envolvendo as três coordenadas paramétricas (ξ, η, ζ)[18].

$$u_i \approx c_1 + c_2\xi + c_3\eta + c_4\zeta + c_5\eta\zeta + c_6\xi\zeta + c_7\xi\eta + c_8\xi\eta\zeta$$

5.1. Sólidos Elásticos 3-D

A partir dos elementos finitos é possível obter características como a matriz de flexibilidade e o módulo de elasticidade do sólido, essenciais para estudos de tensão, dentre outros.

Para o caso de um sólido elástico tridimensional isotrópico e sem qualquer carregamento inicial, a relação entre tensão e deformação é dada por:

$$[\sigma] = \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{yz} \\ \tau_{zx} \\ \tau_{xy} \end{Bmatrix} = [D] \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{yz} \\ \gamma_{zx} \\ \gamma_{xy} \end{Bmatrix} = [D]\{\varepsilon\}$$

Sendo E o módulo de Young e ν o coeficiente de Poisson, temos a matriz de elasticidade [D] do elemento.

$$[D] = \frac{E}{(1-2\nu)(1+\nu)} \cdot \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{(1-2\nu)}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{(1-2\nu)}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{(1-2\nu)}{2} \end{bmatrix}$$

Assim sendo, [D] é a inversa da matriz de flexibilidade [C], e, portanto:

$$\{\varepsilon\} = [C]\{\sigma\}$$

Agora, considerando que o domínio Ω é decomposto em N_E elementos finitos, o total de energia de deformação U do corpo elástico será dado por:

$$U = \sum_{e=1}^{N_E} \frac{1}{2} \int_{\Omega_e} \{\varepsilon\}(u)^T [D]\{\varepsilon\}(u) d\Omega$$

Enquanto a energia relacionada ao trabalho realizado por forças externas (força denominada b e tração denominada t) é dada por:

$$W = \sum_{e=1}^{N_E} \left\{ - \int_{\Omega_e} (u)^T \rho\{b\} d\Omega - \int_{\partial\Omega_e \cap \Gamma_t} (u)^T \{t\} d\Gamma \right\};$$

onde ρ é a densidade do material.

Finalmente, a energia potencial total do sólido é: $F = U + W$.

5.2. Elemento Hexagonal Nodal

De forma geral, um elemento de 8 nós é mapeado comparando-se as suas arestas com as de um cubo original. Assim, considerando as coordenadas físicas do elemento mapeado como sendo x, y e z, o mapeamento pode ser descrito como a seguir:

$$x = \{\ 1 \quad \xi \quad \eta \quad \zeta \quad \xi\eta \quad \eta\zeta \quad \zeta\xi \quad \xi\eta\zeta \} \begin{Bmatrix} c_{x1} \\ c_{x2} \\ c_{x3} \\ c_{x4} \\ c_{x5} \\ c_{x6} \\ c_{x7} \\ c_{x8} \end{Bmatrix} = [b_f]^T \{c_x\}$$

Analogamente, $y = [b_f]^T \{c_y\}$ e $z = [b_f]^T \{c_z\}$.

Da correspondência entre as coordenadas paramétricas e físicas pode ser tirada a seguinte relação para cada nó i do elemento cúbico:

$$\{x_i \quad y_i \quad z_i\} = \{b_f(\xi_i,\eta_i,\zeta_i)^T c_x \quad b_f(\xi_i,\eta_i,\zeta_i)^T c_y \quad b_f(\xi_i,\eta_i,\zeta_i)^T c_z\};\ \text{onde i = 1,...,8.}$$

Através da manipulação simbólica desse resultado, podemos obter as funções de forma do problema (N_1,..., N_8), como também módulo de elasticidade ε do elemento.

$$\begin{Bmatrix} x \\ y \\ z \end{Bmatrix} = \begin{bmatrix} N_1 & 0 & 0 & N_2 & 0 & 0 & \dots & N_8 & 0 & 0 \\ 0 & N_1 & 0 & 0 & N_2 & 0 & \dots & 0 & N_8 & 0 \\ 0 & 0 & N_1 & 0 & 0 & N_2 & \dots & 0 & 0 & N_8 \end{bmatrix} \begin{Bmatrix} x_1 \\ y_1 \\ z_1 \\ x_2 \\ y_2 \\ z_2 \\ \vdots \\ x_8 \\ y_8 \\ z_8 \end{Bmatrix}$$

Voltando à forma básica do elemento original, o volume de cada *Voxel* pode ser descrito como:

$$V = \det[J d\xi d\eta d\zeta];$$

onde [J] é o Jacobiano da transformação isoparamétrica.

E, finalmente, de acordo com o equacionamento desenvolvido no item 4.2.1., a energia total de deformação do elemento é:

$$U_e = \frac{1}{2}\{d_e\}^T [K_e]\{d_e\};$$

onde $\{d_e\}$ é o vetor de deslocamentos e $[K_e]$ é a matriz de elasticidade.

6. VISUALIZAÇÃO DE SÓLIDOS 3-D

Atualmente existem diversos programas dedicados a fornecer ferramentas de visualização tridimensional. No entanto, este projeto está focado no uso do *Visualization Toolkit* (VTK), pois se trata de uma poderosa biblioteca de visualização compatível com linguagens de programação bastante difundidas. Além disso, o VTK tem a vantagem de ser um software *open-source*, eliminando o problema de custo da ferramenta.

Além de oferecer uma boa visualização, o VTK também permite a criação de imagens dinâmicas, onde o usuário pode ter uma visão tridimensional da cena, com diferentes ângulos. A imagem pode ser aproximada, distanciada e rotacionada, dando ao usuário um resultado imediato. Na Figura 9 há a representação de um ensaio de MEF visualizado em VTK, demonstrando que a ferramenta é adequada ao projeto.

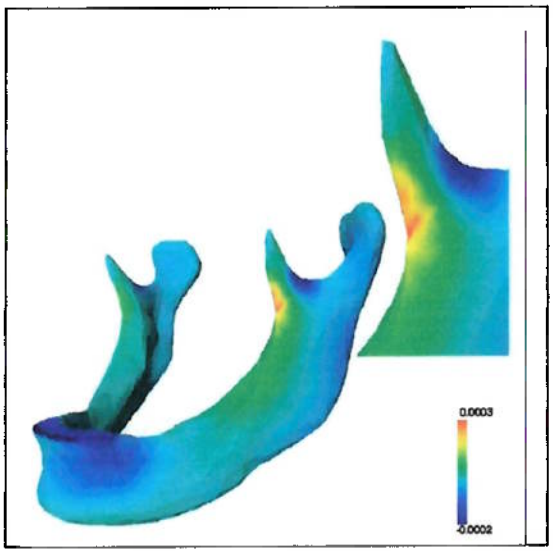


Figura 9 - Simulação de MEF vista em VTK

O processo de construção do modelo digital pode, ainda, ser aperfeiçoado se alguns aspectos forem considerados, tais como:

- A geometria da imagem deve ser conhecida e fixa, ou seja, as dimensões do sólido real e o número de *pixels* das imagens devem ser previamente estabelecidos;
- A sensibilidade dos contornos do sólido deve ser controlada, para evitar que pequenos erros de aquisição das imagens afetem o resultado.

6.1. VTK – Modelo Gráfico

O VTK é constituído por uma série de objetos: Câmera, Luz, Volume, *Render*, dentre outros [15]. Uma combinação desses elementos monta uma "Cena", que será o ambiente visualizado pelo usuário. Um exemplo simples pode ser visto na Figura 10.

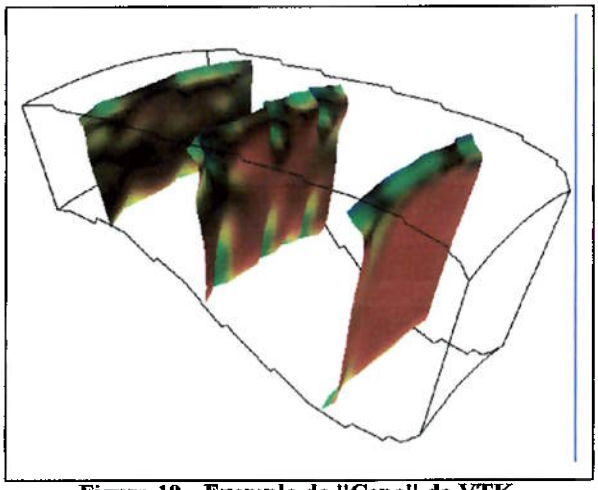


Figura 10 - Exemplo de "Cena" do VTK

Tratando-se de um ambiente orientado a objetos, o anexo A apresenta como os objetos da biblioteca VTK se relacionam computacionalmente. A seguir é apresentada uma breve descrição dos principais objetos gráficos envolvidos na construção de um cenário.

- Prop – Props são todas as coisas que o usuário pode ver na cena. Para imagens tridimensionais a subclasse utilizada é vtkProp3D; para representações bidimensionais usa-se vtkActor2D. Tratando-se das propriedades visíveis do objeto, uma Prop está relacionada à aparência do mesmo, definindo características como cor, opacidade e difusão de luz. Se esse objeto for 3-D, sua Prop também deverá estar relacionada a uma subclasse de transformação interna (vtkTransform), que encapsula uma matriz de transformação 4 x 4 responsável por controlar posição, orientação e escala do objeto.

- Luz – A subclasse *vtkLight* é usada para representar e manipular a luminosidade de um cenário 3-D. Cenários 2-D não possuem essa propriedade.

- Câmera – *vtkCamera* é uma subclasse que determina como uma geometria 3-D deve ser projetada em um plano 2-D. Diversos fatores influenciam essa projeção, como posição, foco e orientação da Câmera. Assim como a Luz, a Câmera não é aplicável a cenários bidimensionais.

- Mapeador – A subclasse *vtkMapper*, em conjunto com a *vtkLookupTable*, é usada para transformar e renderizar uma geometria. O Mapeador permite a interface entre o modelo de visualização e o modelo gráfico do software. Quanto à subclasse *vtkLookupTable*, esta provém da *vtkScalarsColors*, que é responsável por mapear as cores dos elementos, função essencial às técnicas de visualização.

- Renderizador – A interface entre o modelo gráfico e o de visualização é construída com base nas subclasses *vtkRenderer* e *vtkRenderWindow*. Um renderizador é um objeto capaz de desenhar na tela os dados armazenados por um *vtkActor*. Mais de um objeto Renderizador pode atuar em uma janela, bem como se pode criar um Renderizador de múltiplas janelas. Assim, não há uma função bijetora entre janelas e objetos Renderizadores (Figura 11). Para entendimento da lógica de software relacionada à renderização de objetos, deve ser introduzido o conceito de *"viewport"*.

  "Viewport é a região de uma janela onde um renderizador pode desenhar. Cada viewport está relacionada a uma única janela e a um único renderizador."[15]

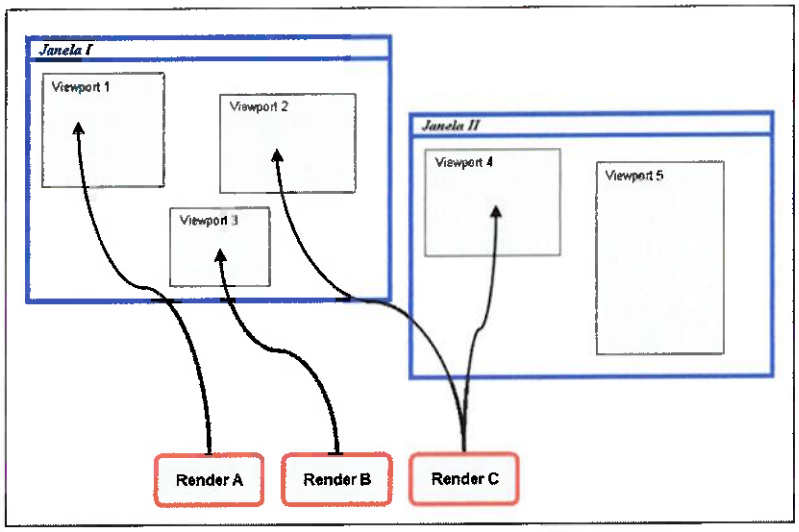


Figura 11 - Relação entre Renderizadores e Janelas

- Integrador – Finalmente, a subclasse *vtkRenderWindowInteractor* permite que o usuário interaja com os objetos desenhados na janela do software. Através dessa ferramenta pode-se manipular a câmera, movimentar os objetos renderizados e alterar algumas propriedades dos atores da cena.

6.2. VTK – Modelo de Visualização

Enquanto o objetivo do modelo gráfico é transformar dados em imagens, o objetivo do modelo de visualização é transformar qualquer tipo de informação em dados gráficos possíveis de serem armazenados pelo computador. Em suma, o modelo de visualização é responsável por construir a representação geométrica que será renderizada pelo modelo gráfico.

Existem dois tipos básicos de objetos envolvidos neste modelo: *vtkDataObject* e *vtkProcessObject*. O primeiro representa dados de diversos tipos, como pontos, vetores e *arrays*. Já o segundo refere-se genericamente a filtros, responsáveis por operar os objetos da *vtkDataObject*, tal como esquematizado na figura a seguir.

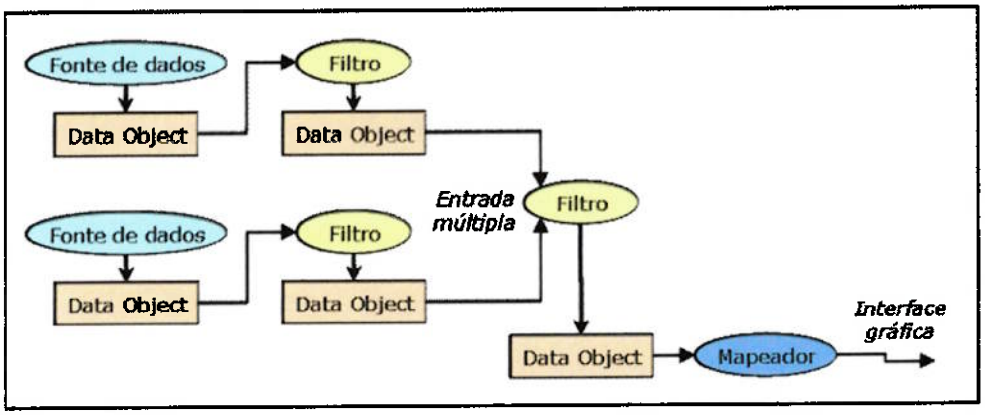


Figura 12 – "Data Objects" conectados por "Process Objects"

7. IMPLEMENTAÇÃO COMPUTACIONAL

7.1. Estruturação do Software

A idéia básica do software é a integração de funções já existentes, constituindo uma ferramenta passível de uso por profissionais das áreas médica e odontológica. O fluxo de informações do software é apresentado na Figura 13. Como se trata de uma integração de diversas funções, cada bloco do software pode ser desenvolvido independentemente, desde que sejam considerados os formatos de comunicação entre os blocos.

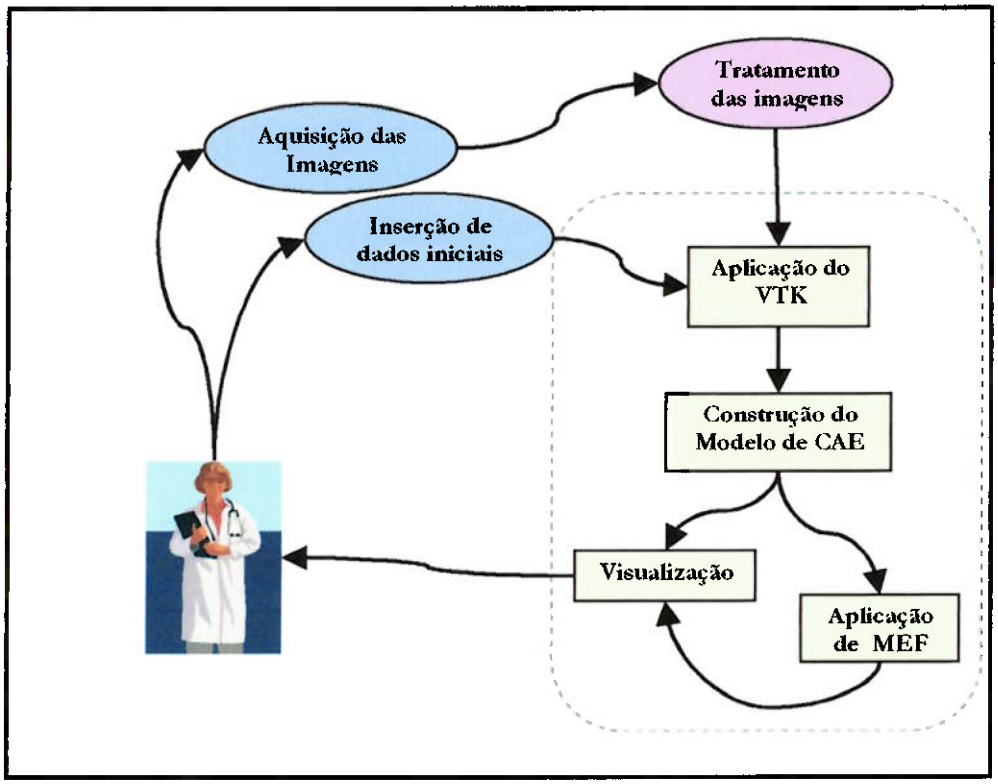


Figura 13 - Funcionamento básico do software

As principais funções constitutivas da aplicação do VTK são: Digitalização das imagens, Discretização, e Reconstrução do sólido tridimensional. O software é complementado com funções de Construção do modelo de CAE, Visualização e Aplicação de MEF.

7.2. Tratamento das Imagens

Antes do início das simulações com o programa, foi necessário fazer um tratamento das imagens obtidas, para que estas se tornassem compatíveis com os conceitos do software. De acordo com a implementação numérica do programa, os *pixels* das imagens podem variar de 0 (branco) a 1 (preto), sendo zero onde há ausência de material. Por outro lado, as imagens de tomografia, assim como as de raios-X, geralmente são obtidas como um negativo; assim, se as imagens fossem usadas diretamente nas simulações, o software compreenderia que o material estaria na parte externa e no centro haveria ausência de material, como pode ser visualizado na figura abaixo.

Para que as imagens sejam compatíveis com o programa elas também devem ser renomeadas, trocando-se a extensão "*.bmp*" por "*.n*", onde n é a numeração da imagem. Por exemplo, se as imagens arquivadas são: *mandibula1.bmp*, *mandibula2.bmp*, *mandibula3.bmp*; elas devem ser renomeadas para: *mandíbula.1*, *mandíbula.2*, *mandíbula.3* e assim por diante.

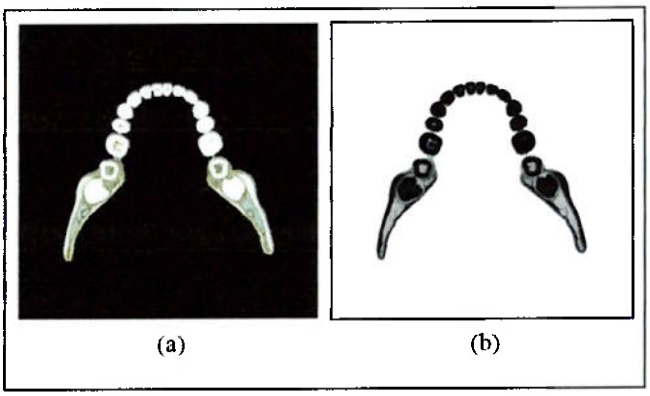


Figura 14 – Tratamento das imagens: (a) antes, (b) depois

7.3. Aplicação do VTK

As funções de digitalização, discretização e reconstrução foram todas implementadas com o auxílio da biblioteca de visualização VTK, como esquematizado na Figura 15. Os passos seguidos foram os seguintes:

1. Leitura dos arquivos de entrada
2. Criação de uma superfície para cada componente anatômico de interesse
3. Transformação das imagens bidimensionais em um modelo tridimensional interativo
4. Renderização das superfícies

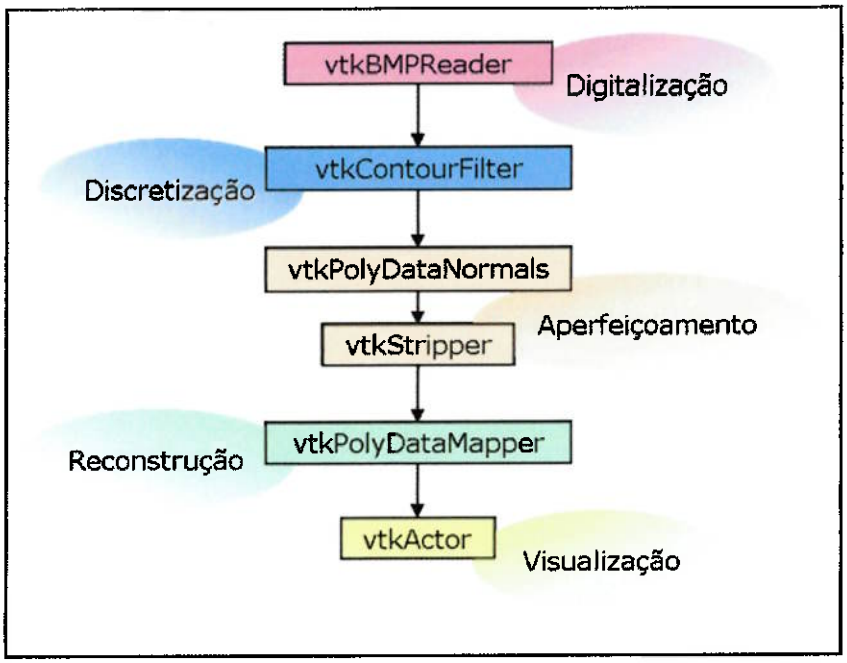


Figura 15 - Diagrama das classes do VTK aplicadas

7.3.1. Leitura dos arquivos de entrada

A biblioteca VTK possui diversas classes que funcionam como leitores de imagens. A classe adequada deve ser escolhida de acordo com o formato da imagem que se deseja ler. Neste projeto, como usualmente o resultado de tomografias são imagens bitmap (extensão BMP), a classe mais adequada é a `vtkBMPReader`.

Fazendo um paralelo com a teoria, esta classe é responsável pela digitalização das imagens, ou seja, ela captura imagens em *pixels* e os transforma em dados passíveis de serem manipulados por outras funções do VTK.

Inicialmente o processo de transformar cada *pixel* da imagem em um valor era feito por um software pré-existente desenvolvido por alunos da Escola Politécnica da USP. O software, designado *"gif2matwin"*, permite que o usuário escolha qualquer imagem do seu computador, e, a partir do arquivo escolhido gera um arquivo de texto contendo uma extensa matriz numérica. *Pixels* da cor branca têm valor zero, e *pixels* da cor preta valem um; qualquer outra cor, inclusive tons de cinza, tem um valor intermediário entre zero e um.

Ao contrário desse software, a classe do VTK não permite que visualizemos a matriz numérica gerada a partir das imagens. Apesar disso sabe-se que o resultado obtido é bastante semelhante, porém com representação em algarismos binários. Embora a biblioteca do VTK seja mais complexa, a possibilidade de se usar uma ferramenta única tornou o seu uso vantajoso em relação ao software anteriormente testado.

7.3.2. Criação de iso-superfícies

As imagens tomográficas fazem distinção entre materiais através das cores, ou seja, cada tom de cinza da imagem corresponde a um material diferente. No entanto, a tomografia não é capaz de capturar informações sobre as características de cada material, como densidade e dureza; portanto, esse conhecimento deve ser prévio.

Uma iso-superfície pode ser definida como uma superfície gerada a partir de pontos com o mesmo potencial, ou seja, com o mesmo valor escalar. Existem três técnicas básicas para construção dessas superfícies: renderização volumétrica, união de cubos (*marching cubes*) e divisão de cubos (*dividing cubes*). A renderização volumétrica é bastante pesada e torna a aplicação lenta; e, portanto não é adequada, pois o software a ser desenvolvido tem a premissa de o usuário poder interagir com os modelos criados. A técnica *marching cubes* permite que a renderização do objeto seja parcialmente feita no hardware, enquanto na técnica dividing cubes a renderização é função somente do software. Assim, o uso da *marching cubes* aumenta consideravelmente as velocidades de processamento do software.

No entanto, a classe do VTK desenvolvida com a técnica de união de cubos, `vtkMarchingCubes`, é patenteada pela General Electric (GE), e não pode ser usada livremente. Segundo a referência *[17]*, em substituição à classe `vtkMarchingCubes` pode ser usada a `vtkContourFilter`. Essa classe funciona como um filtro gerando iso-superfícies a partir dos valores escalares obtidos das imagens pela classe `vtkBMPReader`. Ao criar essas superfícies, a aplicação trata o problema de discretização apresentado na teoria, criando valores intermediários aos dos pontos conhecidos.

Anteriormente havia sido proposto fazer o processo de discretização a partir de um programa já existente desenvolvido em linguagem C. No entanto, a classe do VTK, por ser orientada a objeto, tornou-se muito mais adequada, por reduzir significantemente o tempo computacional da aplicação. Além disso, o programa em C era aplicável somente em situações com um único material, o que não é o caso da anatomia humana (vide apêndice II).

Por exemplo, no caso da mandíbula, há presença de osso, dentina, gengiva, etc; materiais que, apesar de bastante diferentes, estão intrinsecamente ligados. As imagens tomográficas consideram os tons de cinza para representar os diferentes materiais. Porém, o processo de discretização também se baseia em tons de cinza para delimitar o contorno da imagem.

Assim, a idéia básica de como as classes do VTK são capazes de tratar esse problema sem comprometer o fator tempo é a seguinte: há duas aplicações semelhantes trabalhando em paralelo: uma responsável pela forma do objeto e a outra pelas cores, ou seja, pela distribuição dos materiais.

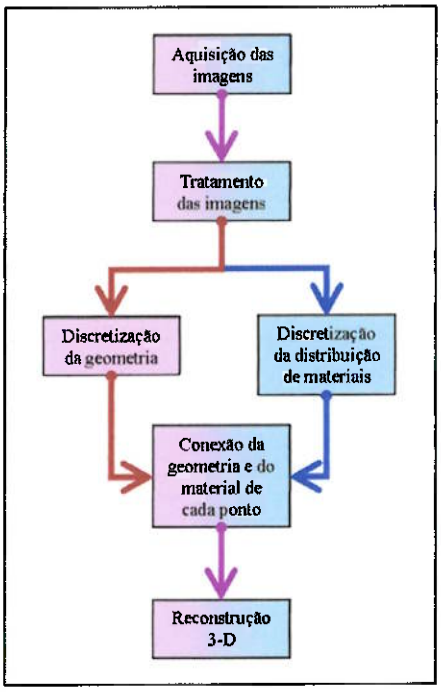


Figura 16 - Fluxograma de funções para o problema de diferentes materiais

No caso da classe `vtkContourFilter`, vale ressaltar que seu desenvolvimento é baseado na classe `vtkShepardMethod`, uma classe básica de aplicação do Método de Shepard descrito anteriormente. Na prática, devem ser criados diversos elementos da `vtkContourFilter`, uma para cada material que se deseja extrair.

7.3.3. Modelo tridimensional interativo

Para se ter uma aplicação do VTK completa é preciso que todos os elementos descritos na fundamentação teórica estejam presentes. Assim, para complementar a aplicação, as seguintes funções são necessárias:

1. Mapeamento dos elementos separadamente
2. Associação de atores a cada mapa criado

A classe `vtkPolyDataMapper` permite associar a um conjunto de pontos um mapa de visualização, que é o que possibilitará a esses dados serem visualizados na tela do computador. Assim como o filtro, essa classe deve ser derivada para cada um dos elementos que se deseja extrair das imagens.

Finalmente, a classe `vtkActor` associa um ator a esse conjunto de pontos. Cada elemento anatômico deve ter o seu próprio ator, pois isto permitirá a manipulação visual dos mesmos. Essa classe permite a manipulação de propriedades como: cor, transparência, visibilidade, etc.

7.3.4. Renderização

A fim de aperfeiçoar o processo de renderização da mandíbula, foram acrescentadas ao programa algumas funções de suavização e performance. A classe `vtkPolyDataNormals` faz com que seja considerada a componente normal média de cada elemento, e a classe `vtkStripper` trata boa parte do ruído. O uso dessas classes torna a transição entre as imagens mais suave, como pode ser observado nas Figuras 17 e 18.

Figura 17 - Renderização sem funções de aperfeiçoamento

Figura 18 - Renderização com funções de aperfeiçoamento

7.4. Aplicação de MEF

Foi incorporada ao software uma rotina de implementação de elementos finitos. Essa rotina, escrita em C++, foi baseada em um programa pré-existente, desenvolvido de acordo com a referência 18, e customizada para os parâmetros envolvidos no problema.

A interface entre a aplicação do VTK e a rotina de MEF foi feita a partir da classe `vtkVoxel`, uma classe capaz de extrair parâmetros de cada *voxel*, como cor e posicionamento na *viewport*. O objetivo básico dessa rotina é transformar os *voxels* formadores do volume tridimensional em elementos cúbicos finitos, compatíveis com as análises baseadas em MEF.

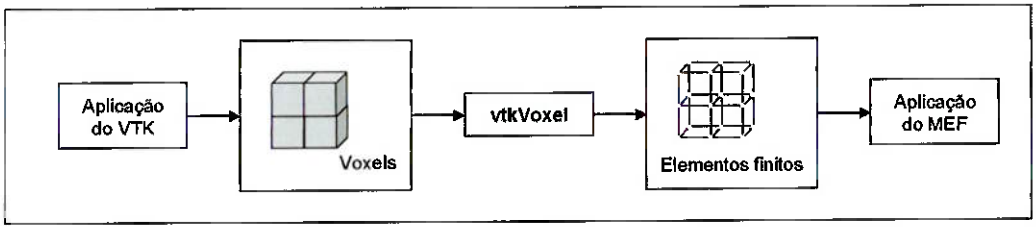


Figura 19 - Ligação entre o VTK e o MEF

Como dito anteriormente, a implementação da aplicação de MEF baseia-se em elementos cúbicos de 8 nós, representados na figura abaixo. O equacionamento desses elementos é detalhado na referência 18.

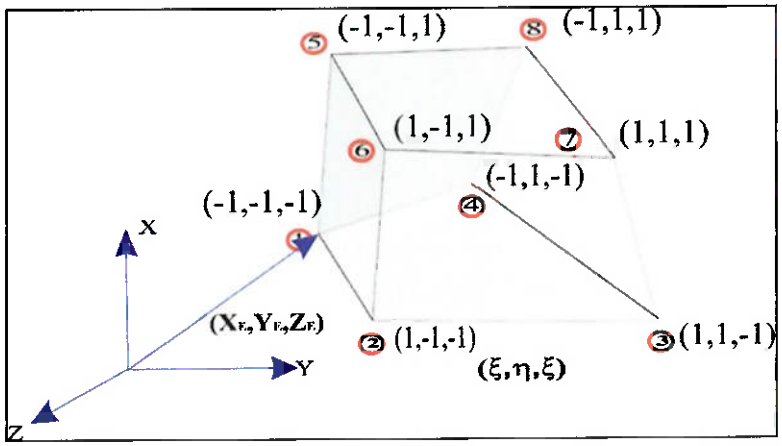


Figura 20 - Elemento Isoparamétrico de 8 nós

7.5. Interface para o programa

Ao executar o software uma janela semelhante à da Figura 21 é aberta. Não é possível obter qualquer visualização sem antes indicar o local onde estão as imagens geradas pelo tomógrafo.

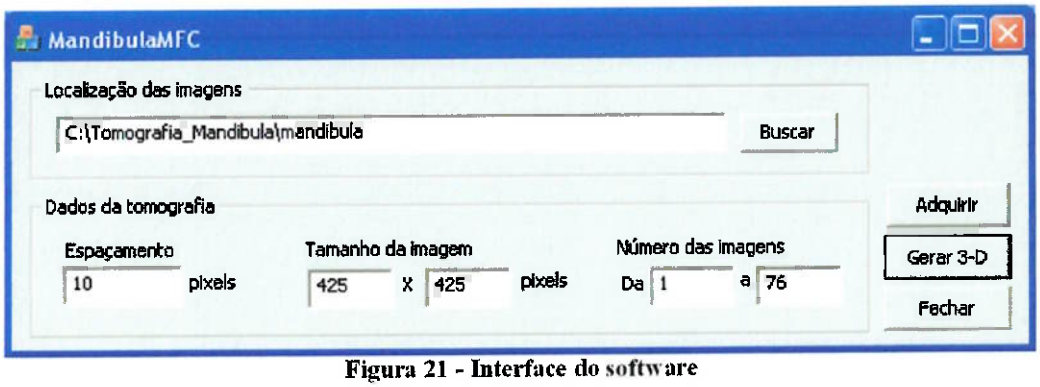


Figura 21 - Interface do software

O usuário também deverá inserir algumas informações básicas sobre a tomografia que deseja reconstruir:

- Espaçamento – é a distância entre as imagens obtidas, ou seja, a resolução da tomografia no eixo ortogonal às imagens;
- Tamanho da imagem – é o número de *pixels* que a imagem bitmap possui;
- Número das imagens – indica qual a seqüência de imagens que deverá ser usada. O usuário não precisa selecionar todas as imagens, pode escolher apenas um intervalo para reconstruir; nesse caso o sólido resultante será uma seção da peça original.

Há duas funções disponíveis no software. O botão adquirir permite que o usuário visualize como as imagens estão sendo capturadas, bem como apresentado na Figura 22, permitindo que o mesmo verifique se as imagens estão corretas.

Figura 22 - Imagens adquiridas

O segundo passo é feito pelo botão "Gerar 3-D", que efetivamente constrói o modelo tridimensional a partir das imagens indicadas. Esse modelo pode ser rotacionado pelo usuário com o auxílio do mouse.

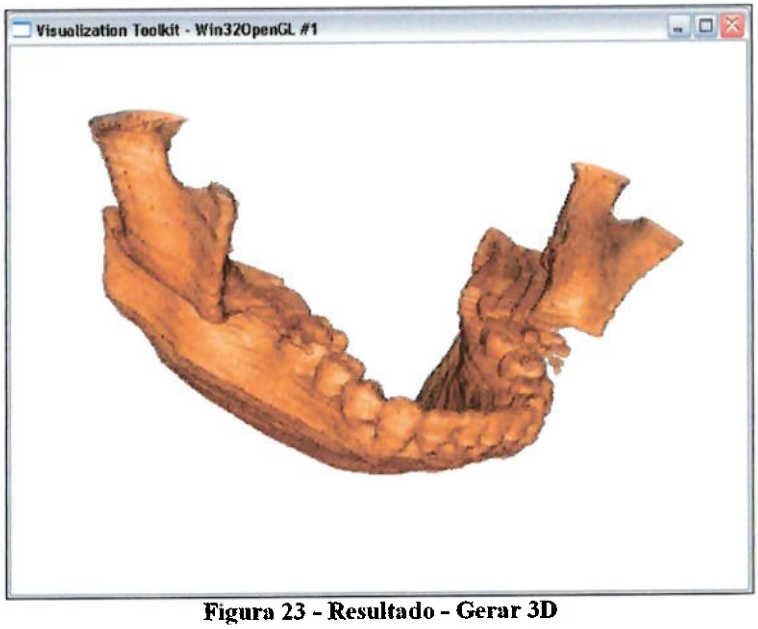


Figura 23 - Resultado - Gerar 3D

8. SIMULAÇÃO

8.1. Simulações do modelo

O modelo simulado é baseado na tomografia de uma mandíbula seca. As imagens das seções transversais da mandíbula foram capturadas em um tomógrafo, tal como descrito anteriormente. Na Figura 24, em preto temos duas estruturas diferentes: a camada óssea externa da mandíbula e a dentina dos dentes; como ambas possuem a mesma cor na tomografia, não será possível tratá-las como estruturas separadas. Assim, o resultado será, basicamente, duas iso-superfícies: uma externa e outra interna. O layout completo das imagens obtidas pode ser visto no anexo B.

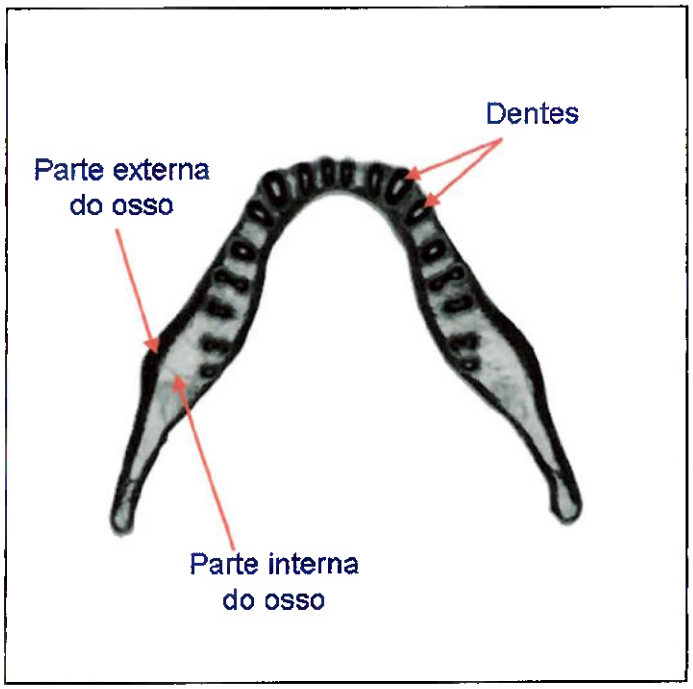


Figura 24 - Detalhes da mandíbula tomografada

Algumas imagens do resultado obtido estão representadas nas figuras abaixo.

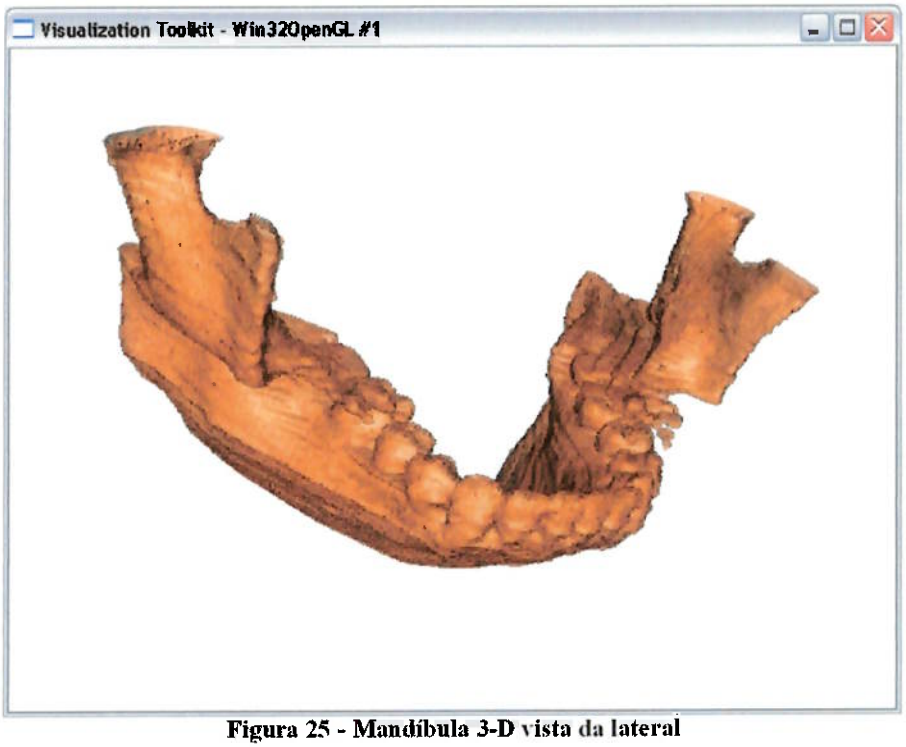


Figura 25 - Mandíbula 3-D vista da lateral

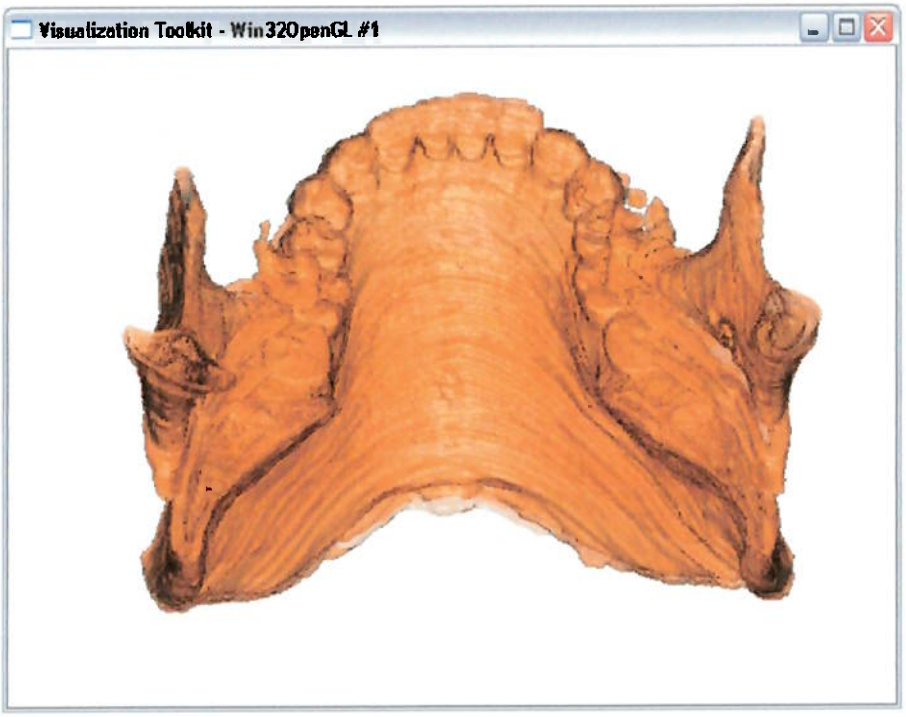


Figura 26 - Mandíbula 3-D vista de trás

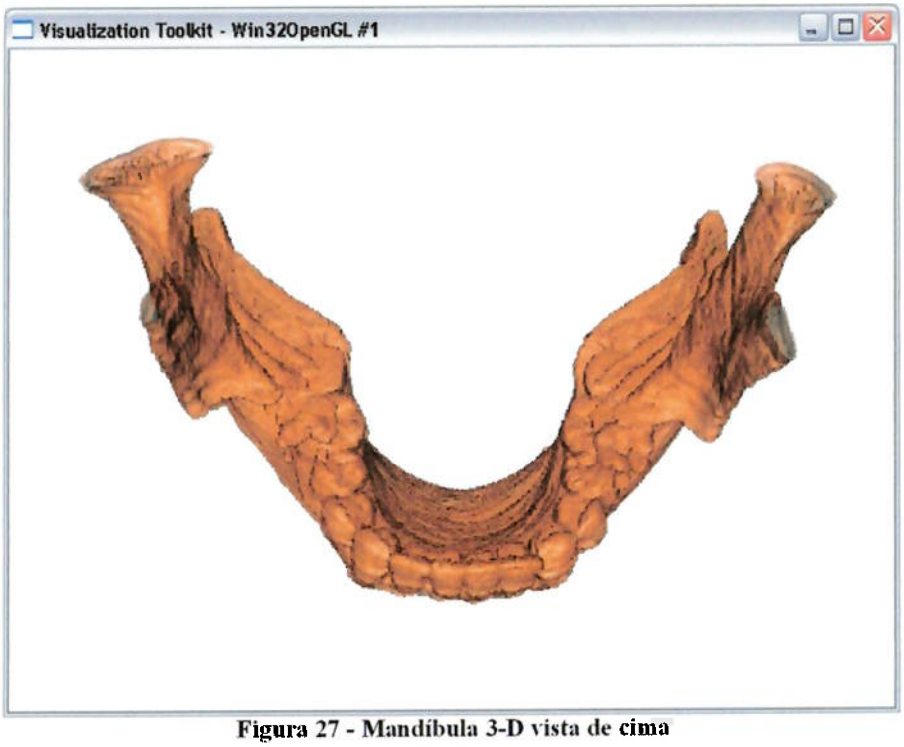


Figura 27 - Mandíbula 3-D vista de cima

Nesse modelo, como se trata de uma mandíbula seca, ou seja, constituída basicamente de osso, não são percebidos diferentes materiais na camada externa do modelo. Isso ocorre porque esta representa ossos e dentes, materiais que se comportam de forma semelhante quando tomografados.

A fim de demonstrar a simulação de diferentes materiais, foi feita a construção de uma seção da mesma mandíbula, conforme figura abaixo. A parte interna do osso, representada pela área cinza nas figuras bitmap, aparece em tons amarelos. Já o que está na cor preta nas figuras bitmap aqui surge como outro material, em tons azulados.

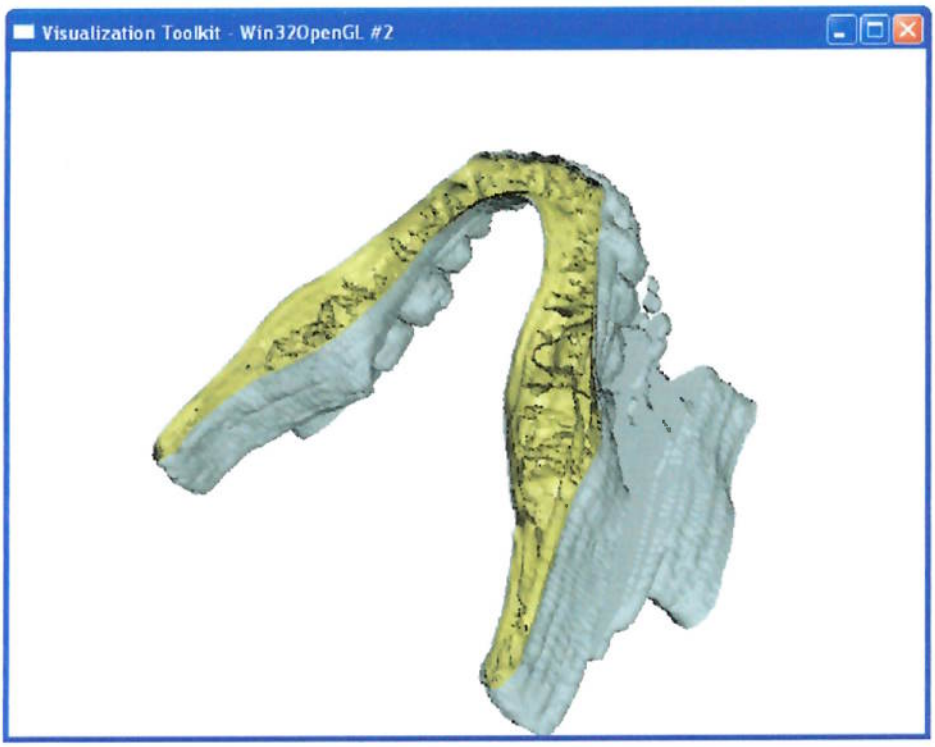


Figura 28 - Construção com diferentes materiais

8.2. Manipulação das imagens

Alterações simples nas imagens bitmap podem gerar grandes alterações no modelo tridimensional. Aqui é apresentado um exemplo de manipulação das imagens, através do qual um dos dentes da mandíbula foi extraído do modelo 3-D. As imagens que sofreram alteração estão identificadas com um asterisco (*) no anexo B. O resultado pode ser visualizado na figura 29.

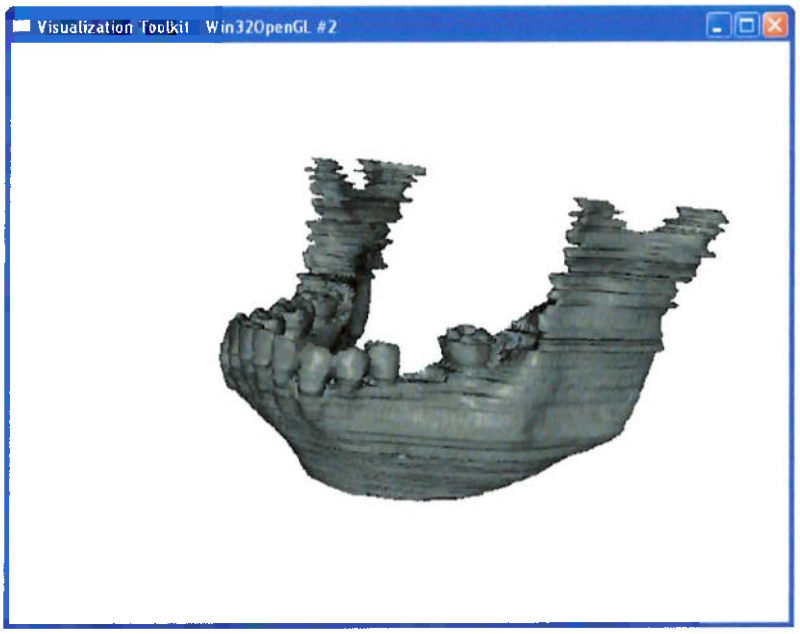


Figura 29 - Resultado da manipulação das imagens

9. CONCLUSÕES

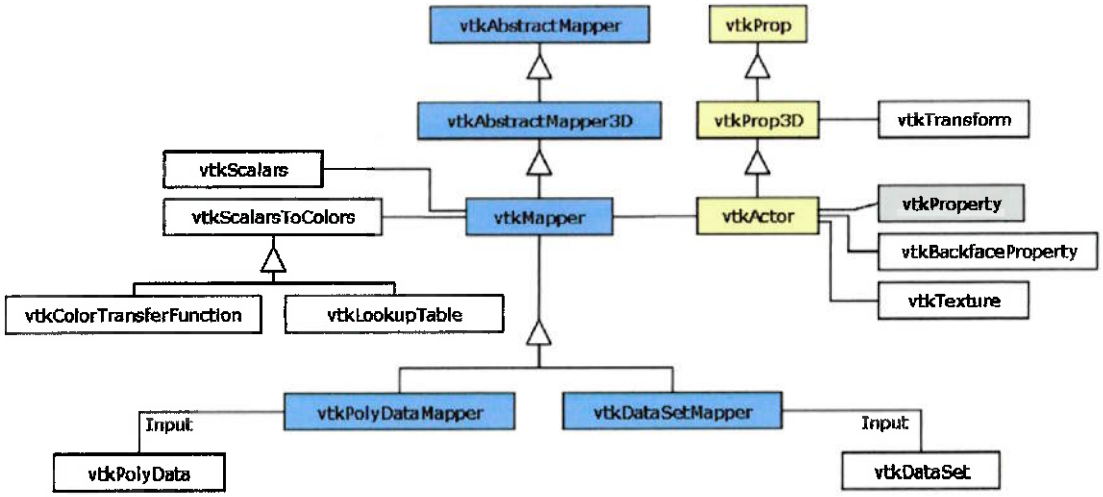
Os seguintes pontos foram abordados: a criação de um protótipo para a interface com o usuário, a incorporação de técnicas de MEF ao software e também a integração dos projetos já existentes em uma linguagem única, C++.

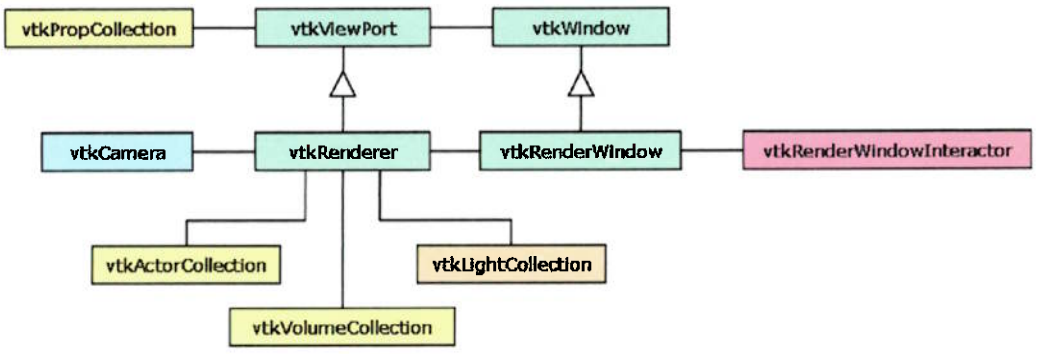
Como modelo de simulação do software, foi usada uma mandíbula humana, cujas imagens de tomografia encontram-se no anexo B. Trata-se de um exemplo completo, pois engloba tanto o problema de diversos materiais como também as técnicas de MEF mais adequadas à área odontológica.

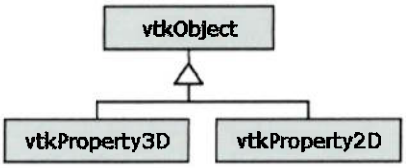
Os resultados apresentaram um bom desempenho do software, pois o tempo computacional foi reduzido substancialmente, além de ter sido obtida uma excelente fidelidade dos modelos em relação aos sólidos originais. No entanto, alguns pontos ainda devem ser desenvolvidos a fim de tornar o software viável, como uma implementação interativa das aplicações de MEF e possibilidades de separação dos elementos anatômicos por parte do usuário.

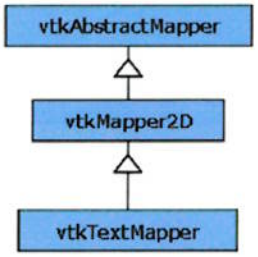
10. ANEXOS

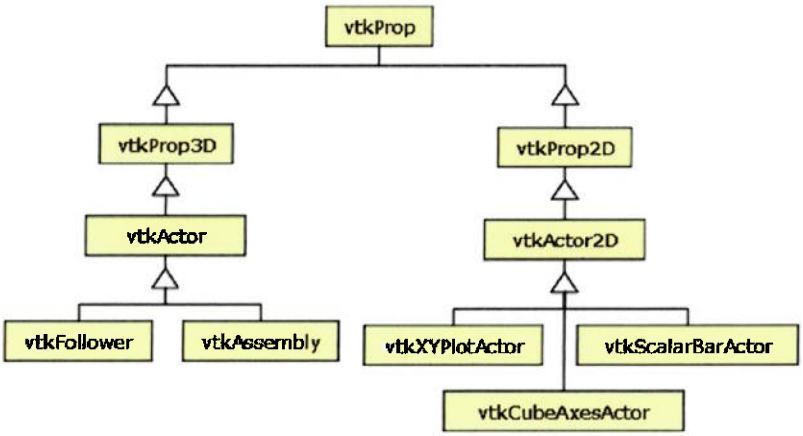
ANEXO A – Diagrama de Classes do Modelo Gráfico da biblioteca VTK



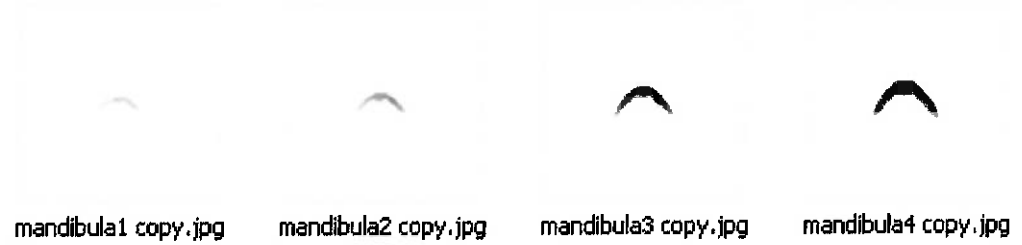



vtkObject
vtkProperty3D
vtkProperty2D


vtkAbstractMapper
vtkMapper2D
vtkTextMapper


vtkProp
vtkProp3D
vtkProp2D
vtkActor
vtkActor2D
vtkFollower
vtkAssembly
vtkXYPlotActor
vtkScalarBarActor
vtkCubeAxesActor

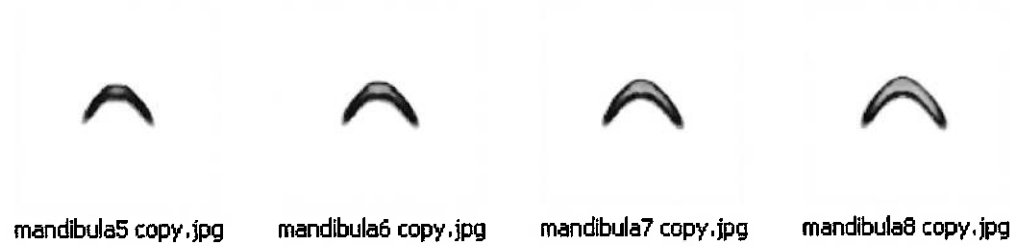
ANEXO B - Conjunto das imagens usadas na reconstrução da mandíbula


mandibula1 copy.jpg

mandibula2 copy.jpg

mandibula3 copy.jpg

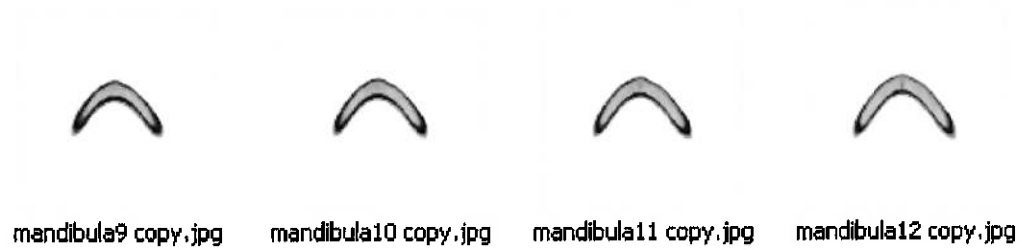
mandibula4 copy.jpg


mandibula5 copy.jpg

mandibula6 copy.jpg

mandibula7 copy.jpg

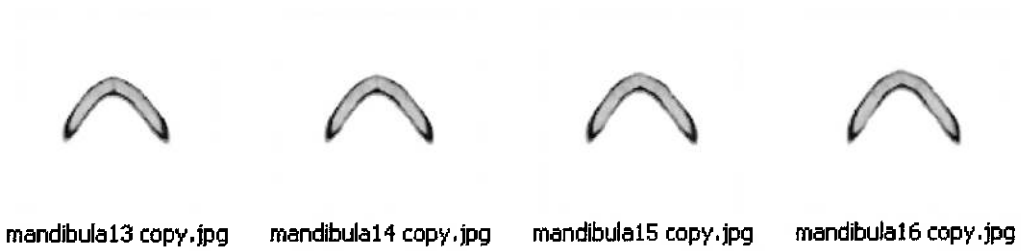
mandibula8 copy.jpg


mandibula9 copy.jpg

mandibula10 copy.jpg

mandibula11 copy.jpg

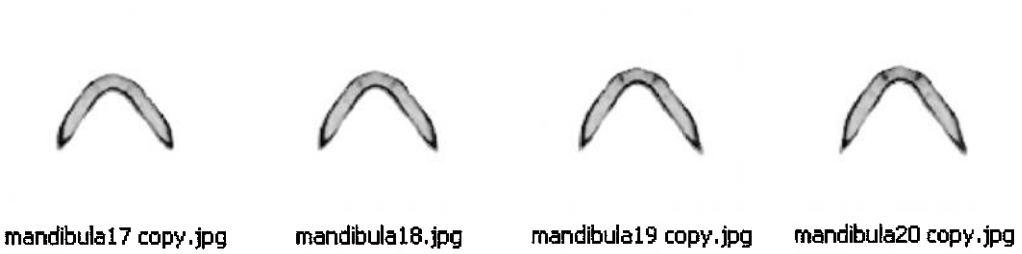
mandibula12 copy.jpg


mandibula13 copy.jpg

mandibula14 copy.jpg

mandibula15 copy.jpg

mandibula16 copy.jpg


mandibula17 copy.jpg

mandibula18.jpg

mandibula19 copy.jpg

mandibula20 copy.jpg

mandibula21 copy.jpg

mandibula22 copy.jpg

mandibula23 copy.jpg

mandibula24 copy.jpg

mandibula25.jpg

mandibula26 copy.jpg

mandibula27 copy.jpg

mandibula28 copy.jpg

mandibula29 copy.jpg

mandibula30 copy.jpg

mandibula31 copy.jpg

mandibula32 copy.jpg

mandibula33 copy.jpg

mandibula34 copy.jpg

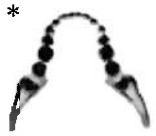
mandibula35 copy.jpg

mandibula36 copy.jpg

mandibula37 copy.jpg

mandibula38 copy.jpg

mandibula39 copy.jpg

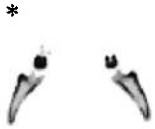


mandibula40 copy.jpg

mandibula41 copy.jpg

mandibula42 copy.jpg

mandibula43 copy.jpg

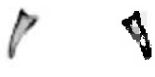


mandibula44 copy.jpg

mandibula45 copy.jpg

mandibula46 copy.jpg

mandibula47 copy.jpg



mandibula48 copy.jpg

mandibula49 copy.jpg

mandibula50 copy.jpg

mandibula51 copy.jpg

mandibula52 copy.jpg

mandibula53 copy.jpg

mandibula54 copy.jpg

mandibula55 copy.jpg

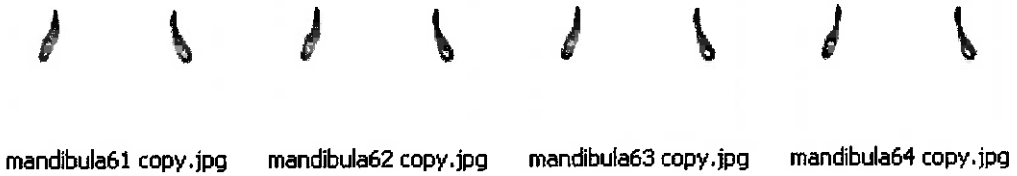
mandibula56 copy.jpg

mandibula57 copy.jpg

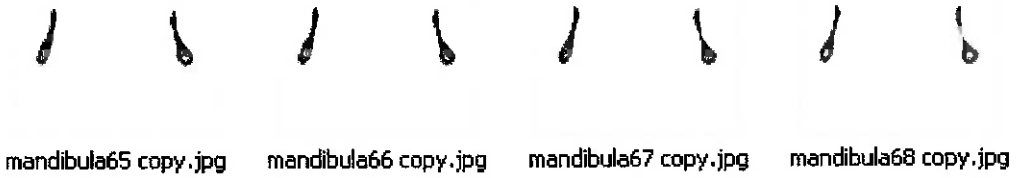
mandibula58 copy.jpg

mandibula59 copy.jpg

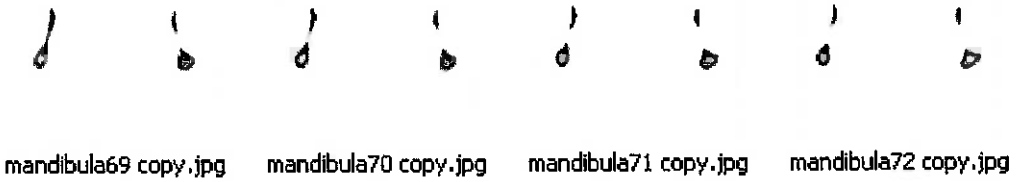
mandibula60 copy.jpg



mandibula61 copy.jpg mandibula62 copy.jpg mandibula63 copy.jpg mandibula64 copy.jpg



mandibula65 copy.jpg mandibula66 copy.jpg mandibula67 copy.jpg mandibula68 copy.jpg



mandibula69 copy.jpg mandibula70 copy.jpg mandibula71 copy.jpg mandibula72 copy.jpg

mandibula73 copy.jpg mandibula74 copy.jpg mandibula75 copy.jpg mandibula76 copy.jpg

11. REFERÊNCIAS BIBLIOGRÁFICAS

[1] Watanabe F., Hata Y., Komatsu S., Ramos T.C., Fukuda H., "Finite element analysis of the influence of implant inclination loading position, and load direction on stress distribution", Department of Crown and Bridge Prosthodontics, The Nippon Dental University School of Dentistry at Niigata, Japan, 2003.

[2] Lin S., Shi S., LeGeros R. Z., LeGeros J. P., "Three-dimensional finite element analysis of four designs of a high-strength silicon nitride implant", Calcium Phosphate Research Laboratory, New York University College of Dentistry, New York, NY, USA, 2003.

[3] Nagasao T., Kobayashi M., Tsuchiya Y., Kaneko T., Nakajima T., "Finite element analysis of the stresses around endosseous implants in various reconstructed mandibular models", Department of Plastic and Reconstructive Surgery, Keio University Hospital, Tokyo, Japan, 2002.

[4] Papavasiliou G., Kamposiora P., Bayne S. C., Felton D.A., "Three-dimensional finite element analysis of stress-distribution around single tooth implants as a function of bony support, prosthesis type, and loading during function", School of Dentistry, University of North Carolina, Chapel Hill, USA, 1996.

[5] Nagasao T., Kobayashi M., Tsuchiya Y., Kaneko T., Nakajima T., "Finite element analysis of the stresses around fixtures in various reconstructed mandibular models – part II (effect of horizontal load)", Department of Plastic and Reconstructive Surgery, Keio University Hospital, Tokyo, Japan, 2003.

[6] Meijer H. J., Starmans F.J., Steen W.H., Bosman F., "A three-dimensional finite element study on two versus four implants in an edentulous mandible", Department of Oral and Maxillofacial Surgery and Maxillofacial Prosthet, University Hospital Groningen, The Netherlands, 1994.

[7] Reis T. C. A., Elias C. N., Gouvêia J. P., "Uso de elementos finitos na análise de tensões no hexágono externo de implantes", Universidade Federal Fluminense, Niterói, RJ, Brasil.

[8] Hannas A. R., Cornacchia T. P. M., Lanza M. D., Las Casas E. B., Carvalho F. F., Reis N. A., "Análise computacional de cavidades classe V em pré-molar

considerando preparo cavitário e materiais restauradores", Universidade Federal de Minas Gerais, MG, Brasil.

[9] Sergio E. M. R., "Método de digitalização de sólidos aplicado ao CAD/CAE", Relatório final de iniciação científica FAPESP, 2000, (processo nº 99/10070-7).

[10] Brauer, H., Ziolkowski, M., Kuilelov, S. Men, Resagk, C., "Surface Current Reconstruction Using Magnetic field Tomography", IEEE Transactions on Magnetics, vol. 40, nº 2, March 2004.

[11] Templeton, Alistair K., Liebschner, Michael A. K., "Construction of Finite Element Vertebral Models from X-Rays", Department of Bioengineering, Rice University, Houston, Texas, 2002.

[12] Lötjönen, J., "Construction of Boundary Element Models in Bioelectromagnetism", Laboratory of Biomedical Engineering, Helsinki University of Technology, Espoo, Finland, 2000.

[13] Kikuchi, N., "Three Dimensional Solid Element – HEXA 8", Computacional Mechanics Laboratory.

[14] Emílio C. N. S., "Apostila de PMR 2420 – Mecânica Computacional", Escola Politécnica da Universidade de São Paulo, São Paulo, 2003.

[15] Schroeder, William J., Martin, Kenneth M., Avila, Lisa S., Law, C. Charles, "The VTK User's Guide", Kitware, Inc., 1998-2000.

[16] Attardi, G., Betrò, M., Forte, M., Gori, R., Guidazzoli, A., Imboden S., Mallegni, F., "3D facial reconstruction and visualization of ancient Egyptian mummies using spiral CT data - Soft tissues reconstruction and textures application", Università di Pisa, Italia.

[17] Schroeder, W., Martin, K., Lorensen, B., "The Visualization Toolkit – An Object-Oriented Approach to 3D Graphics", Prentice Hall, 2nd Edition, 1998.

[18] James Edward Shooter

[19] Bathe, K. J., "Finite element procedures in Engineering Analysis", Prentice-Hall, Inc., Englewood Clifs, New Jersey, 07632, USA, 1982.

12. APÊNDICES

APÊNDICE I – Código-fonte parcial comentado

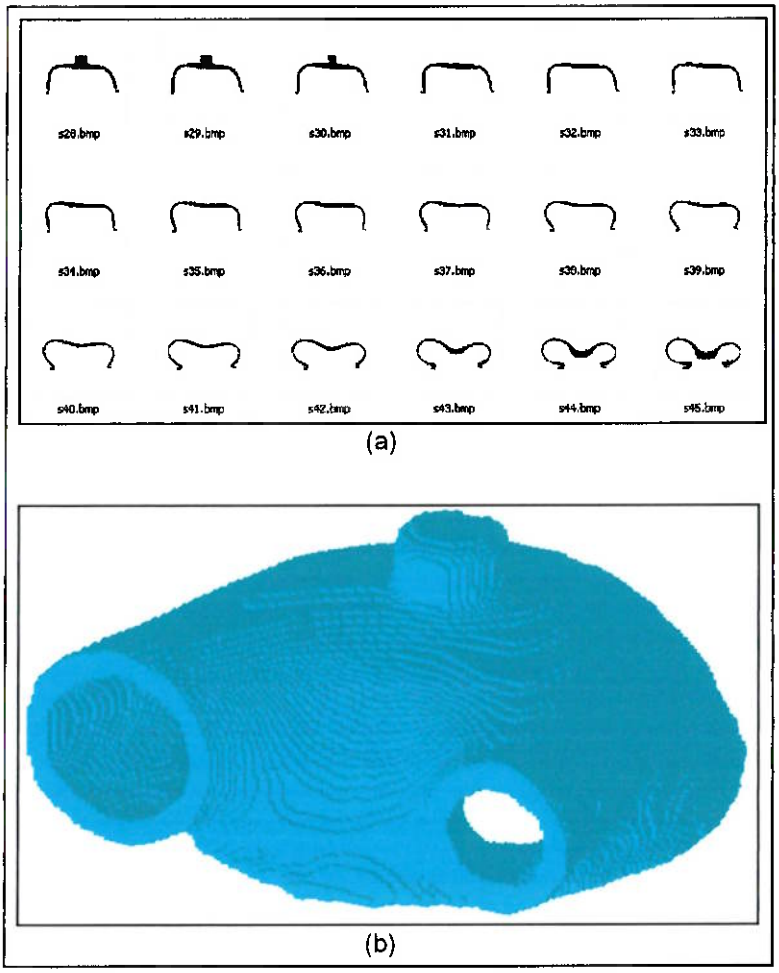
```
/*=========================================================================

      PMR 2550 - Projeto de Formatura II

      Aluna:            Heloisa Alves Fonseca
(heloisa.fonseca@poli.usp.br)
      Orientador: Prof. Dr. Emilio Carlos Nelli Silva

      Arquivo:    OdontoMEF.cpp
      Data:       02/12/2004

      Descrição:
      Este programa lê imagens em formato BMP de 8 bits (256 cores)
no formato:
      PATH\imagem.%d

      Onde:
      PATH é o caminho do diretorio das imagens.
      %d numero decimal variando de 1 até o numero de imagens
(configurável)

      Em seguida e montado um ator com os dados filtrados pelas
rotinas do
      VTK de extração de iso-superficies.


=========================================================================*/


#include <vtkRenderer.h>
#include <vtkRenderWindow.h>
#include <vtkRenderWindowInteractor.h>
#include <vtkVolume16Reader.h>
#include <vtkBMPReader.h>
#include <vtkPolyDataMapper.h>
#include <vtkActor.h>
#include <vtkOutlineFilter.h>
#include <vtkCamera.h>
#include <vtkStripper.h>
#include <vtkLookupTable.h>
#include <vtkImageDataGeometryFilter.h>
#include <vtkProperty.h>
#include <vtkPolyDataNormals.h>
#include <vtkContourFilter.h>
#include <vtkImageData.h>
#include <vtkImageMapToColors.h>
#include <vtkImageActor.h>
#include <vtkVolume.h>

#define BMP_READER_FILE_PREFIX "C:\\Tomografia Mandibula\\imagem"

#define TAMANHO_IMAGEM_PIXELS_X 425
#define TAMANHO_IMAGEM_PIXELS_Y 425
```

```
#define NUMERO_DE_CAMADAS 76
#define NUMERO_DA_PRIMEIRA_CAMADA 1

#define ESPACAMENTO_X 3.2
#define ESPACAMENTO_Y 3.2
#define ESPACAMENTO_Z 10

#define WINDOW_SIZE_X 640
#define WINDOW_SIZE_Y 480

int main (int argc, char **argv)
{
      // Cria a interface com o sistema operacional:
      // Janela, eventos de mouse e teclado, etc...
      vtkRenderer *aRenderer = vtkRenderer::New();
      vtkRenderWindow *renWin = vtkRenderWindow::New();
            renWin->AddRenderer(aRenderer);
      vtkRenderWindowInteractor *iren =
vtkRenderWindowInteractor::New();
            iren->SetRenderWindow(renWin);

      // Lê as seções (imagens BMP) para compor o volume
      // Define o espaçamento dos pixels em coordenada de mundo.
      vtkBMPReader *BMPReader = vtkBMPReader::New();
      BMPReader->SetDataByteOrderToLittleEndian();
      BMPReader->SetDataExtent(0, TAMANHO_IMAGEM_PIXELS_X,
TAMANHO_IMAGEM_PIXELS_Y, 79, NUMERO_DA_PRIMEIRA_CAMADA,
NUMERO_DE_CAMADAS);
      BMPReader->SetFilePrefix(BMP_READER_FILE_PREFIX);
      BMPReader->SetDataSpacing(ESPACAMENTO_X, ESPACAMENTO_Y,
ESPACAMENTO_Z);

      // Seleciona a iso-superfície de valor 100
      // Alternativamente é possível gerar um conjuto de iso-
superfícies
      // igualmente separadas dentro de um intervalo com
GenerateValues.
      vtkContourFilter *mandibulaContour = vtkContourFilter::New();
            mandibulaContour->SetInput(BMPReader->GetOutput());
            mandibulaContour->SetValue(1, 100);
      //contour->GenerateValues(10, 1, 150);

      vtkPolyDataNormals *mandibulaNormals =
vtkPolyDataNormals::New();
            mandibulaNormals->SetInput(mandibulaContour-
>GetOutput());
            mandibulaNormals->SetFeatureAngle(60.0);
      vtkStripper *mandibulaStripper = vtkStripper::New();
            mandibulaStripper->SetInput(mandibulaNormals-
>GetOutput());
      vtkPolyDataMapper *mandibulaMapper = vtkPolyDataMapper::New();
            mandibulaMapper->SetInput(mandibulaStripper-
>GetOutput());
            mandibulaMapper->ScalarVisibilityOff();

      // Faz o ator. Define propriedades como cor, brilho, etc...
      vtkActor *mandibula = vtkActor::New();
            mandibula->SetMapper(mandibulaMapper);
```

```
            mandibula->GetProperty()->SetDiffuseColor(0.9, 0.9,
0.9);
            mandibula->GetProperty()->SetSpecular(.3);
            mandibula->GetProperty()->SetSpecularPower(120);

      // Posiciona a camera em uma posição padrão.
      vtkCamera *aCamera = vtkCamera::New();
            aCamera->SetViewUp(0, 0, 1);
            aCamera->SetPosition(0, 1, 0);
            aCamera->SetFocalPoint(0, 0, 0);
            aCamera->ComputeViewPlaneNormal();

      aRenderer->AddActor(mandibula);
      aRenderer->SetActiveCamera(aCamera);
      aRenderer->ResetCamera();
      aCamera->Dolly(1.5);

      // Define a cor do fundo (branco) e o tamanho da janela.
      aRenderer->SetBackground(1, 1, 1);
      renWin->SetSize(640, 480);

      // Finalmente executa o loop "infinito" que é interrompido ao
      // fechar o programa.
      iren->Initialize();
      iren->Start();

      // Libera a memória alocada para os objetos evitando o
vazamento
      // de memória.
      BMPReader->Delete();
      mandibulaContour->Delete();
      mandibulaNormals->Delete();
      mandibulaStripper->Delete();
      mandibulaMapper->Delete();
      mandibula->Delete();
      aCamera->Delete();
      aRenderer->Delete();
      renWin->Delete();
      iren->Delete();

      return 0;
}
```

APÊNDICE II – Funcionamento do programa em C para discretização

Simulação de uma cavidade de coração artificial formada por apenas 1 material
(a) exemplo de seções transversais
(b) sólido reconstruído visto no *Ansys*



APÊNDICE III – Simulação da cavidade apresentada do apêndice II com o software apresentado neste projeto para fins de comparação.