Uma ferramenta de processamento de imagens médicas dentro do Sistema HeMoLab

Vinicius Pessôa, Ignacio Larrabide, Raul A. Feijóo
LNCC – Laboratório Nacional de Computação Científica
Rua Getúlio Vargas 333 - Petrópolis - RJ
{vpessoa@lncc.br, nacho@lncc.br, feij@lncc.br}

Abstract. *The continuous raising numbers of cardiovascular accidents and the growing complexity of surgical techniques to treat them combined with the fast advance of computational modeling and simulation became a big motivation for the development of computational tools that are capable of predicting the result of these medical procedures using precise and non-invasive methods. In this paper are discussed different tools that, using a distributed architecture, allow the user to incorporate patient specific information from medical images to these models by means of image processing and visualization techniques.*

Resumo. *O aumento do número de incidências de doenças cardiovasculares e da complexidade das intervenções cirúrgicas para tratá-las, aliado ao crescente avanço das técnicas de modelagem e simulação computacional, tem motivado o desenvolvimento de sistemas computacionais capazes de realizar predições dos resultados destas intervenções de maneira precisa e não invasiva. Neste trabalho, são discutidas diferentes ferramentas que, inclusive em modo distribuído, permitem ao usuário incorporar em modelos computacionais informações específicas dos pacientes a partir de imagens médicas, utilizando técnicas de processamento e visualização de imagens.*

1. Introdução

O aumento do número de incidências de doenças cardiovasculares nos últimos anos [1] tem motivado o desenvolvimento de técnicas específicas para o tratamento destas doenças, sendo a implantação de *stents*, *clips* e *bypass* alguns exemplos. A complexidade das intervenções cirúrgicas citadas tem crescido consideravelmente nos últimos anos, devido principalmente ao rápido avanço da medicina moderna. Neste contexto, surge a necessidade de desenvolver ferramentas que permitam predizer com precisão e de forma não invasiva os resultados destas intervenções. Paralelamente, o rápido e constante avanço da modelagem e simulação computacional fornece estas ferramentas, permitindo estudar de forma segura o resultado dos procedimentos médicos acima referenciados [2]. Com o intuito de reunir estas ferramentas em um mesmo ambiente computacional surgiu o HeMoLab – Laboratório de Modelagem em Hemodinâmica [3]. O HeMoLab é um sistema computacional que permite a criação de modelos especializados do SCVH - Sistema Cardiovascular Humano [4, 5], que ajudarão na simulação e predição dos resultados de importantes intervenções cirúrgicas relacionadas ao SCVH utilizando informações específicas de pacientes.

Neste trabalho é descrito o módulo de processamento de imagens do HeMoLab, o qual envolve uma série de leitores de imagens médicas em diferentes formatos, filtros de processamento de imagens, *widgets* para manipulação e visualização de volumes de dados e, por fim, a geração de modelos tridimensionais a partir destes dados. Cabe ressaltar que o intuito deste trabalho é apresentar a ferramenta de processamento de imagens do sistema HeMoLab, que envolve desde a integração entre filtros de processamento de imagens e ferramentas de visualização à utilização de sistemas distribuídos e processamento paralelo. Assim, neste trabalho serão abordadas principalmente questões referentes a arquitetura, padrões de comunicação, estrutura de conexão de filtros e funcionamento do *software* de forma distribuída, não havendo pretensão de apresentar qualquer avaliação sobre os resultados dos filtros de processamento de imagens.

2. Arquitetura

Diferentes ferramentas e *frameworks* foram utilizados como base para desenvolver o HeMoLab, sendo os mais importantes o *software* de visualização de dados ParaView [6] e a biblioteca de visualização tridimensional VTK (*Visualization ToolKit*) [7].

O ParaView é um software *open-source*, baseado em uma arquitetura cliente-servidor, projetado e desenvolvido especificamente para tratar e visualizar grandes volumes de dados. Este pode ser executado em maneira distribuída ou em um único processo. Sua estrutura é baseada na biblioteca VTK, que permite a leitura, conversão e escrita de arquivos em diferentes formatos, filtros de processamento de dados e renderização (i.e., visualização 3D). Para a construção de interfaces, o ParaView emprega Tcl/Tk com ANSI-C++, utilizando *wrappers* para realizar as conversões em tempo de compilação.

O VTK é uma das mais utilizadas bibliotecas de visualização tridimensional, sendo esta *open-source* e desenvolvida sobre OpenGL [8]. Esta ferramenta inclui geração, processamento e visualização de imagens e malhas tridimensionais de vários tipos. Seu processamento pode ser resumido em três etapas:

- Leitura/geração de estruturas de dados;
- *Pipeline* de filtros;
- Criação de atores e renderização 3D.

O VTK consiste basicamente em um conjunto de bibliotecas C++ com interfaces para Tcl/Tk, Java e Python. É importante ressaltar que o VTK implementa leitores de vários formatos de arquivos amplamente utilizados, como leitores de imagens PNG (*Portable Network Graphics*), DICOM (*Digital Imaging and Communications in Medicine*) e arquivos STL (*Stereo Lithography Files*), utilizados respectivamente para o armazenamento de imagens 2D, 3D e malhas 3D. Possui também diversos conversores de estruturas de dados externas para as estruturas por ele utilizadas e exportadores para outros formatos suportados. O VTK oferece ao desenvolvedor uma completa hierarquia de classes, permitindo a este implementar algoritmos próprios em forma de leitores, escritores, filtros, *widgets*, entre outros. Resumindo, em conjunto com o VTK, o ParaView é uma aplicação distribuída que executa aplicações VTK em processos servidores, oferecendo uma interface (API – *Application Programming Interface*) para a criação de fontes de dados (*sources* na forma de leitores, por exemplo), conexão de filtros (gerenciamento do *pipeline*) e visualização (*rendering*). O módulo de processamento de imagens do HeMoLab utiliza toda esta estrutura para dar poder de

armazenamento a suas estruturas de dados e velocidade a seus leitores, filtros e escritores.

Os softwares ParaView e VTK já oferecem um completo conjunto de leitores, escritores, filtros e *widgets*, fornecendo também padrões de comunicação e processamento de possível reutilização. Neste sentido, o sistema HeMoLab pode ser classificado como uma extensão do ParaView, oferecendo novas funcionalidades que permitem a criação, edição e simulação de modelos do Sistema Cardiovascular Humano. Utilizando as ferramentas do módulo de processamento de imagens do HeMoLab, informações específicas de pacientes - fornecidas na forma de imagens médicas - podem ser incorporadas a estes modelos.

A idéia de estender o ParaView originou-se de uma análise de suas funcionalidades aliadas ao grande poder de processamento oferecido pelo VTK. A arquitetura distribuída do ParaView, sua robustez decorrente do fato de ser completamente desenvolvido em ANSI-C++ e sua versatilidade, fizeram dele a melhor escolha para servir como base de desenvolvimento para o HeMoLab. Esta arquitetura permite que o HeMoLab utilize processadores (CPUs) dedicados ao processamento dos dados por meio de filtros e renderização, enquanto o cliente (interface de usuário) acessa estas funcionalidades, seja no mesmo computador ou de forma remota. Existem três componentes principais nesta arquitetura:

- **Cliente** – aplicação que possui a interface de usuário e os principais recursos de interatividade com ele;
- ***Render-Server*** – servidores responsáveis pela função de renderização;
- ***Data-Server*** – servidores responsáveis pelo armazenamento, processamento e conversão dos dados através dos leitores, filtros e escritores.

A Figura 1 ilustra o caminho dos dados na comunicação entre os principais componentes do HeMoLab.

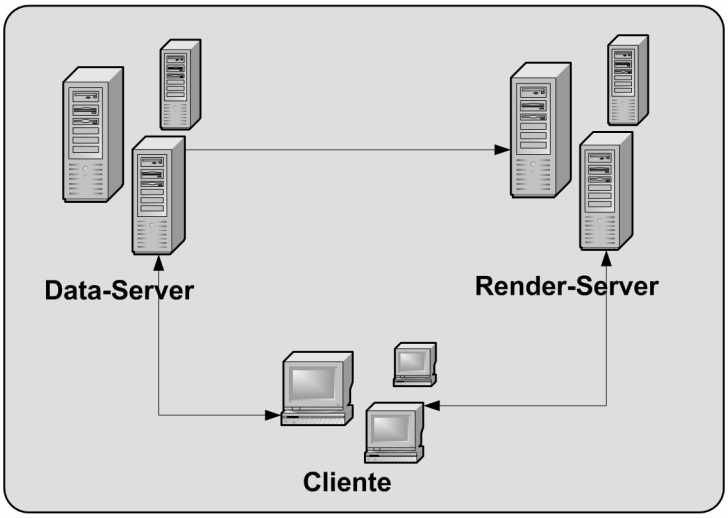

Figura 1. Arquitetura Distribuída do HeMoLab

Quando executado em modo *stand-alone* (i.e., cliente e servidores dentro de um único processo) esta arquitetura torna-se completamente transparente, já que o usuário executa apenas um único aplicativo. No modo distribuído, pode-se criar um ambiente com N *data-servers* e M *render-servers* sendo executados de forma dedicada em até N+M computadores, onde o processo cliente acessará os processos servidores através de uma rede (preferencialmente de alta velocidade) ou de memória compartilhada em arquiteturas de *hardware c*om múltiplos processadores que assim o permitam. É

interessante ressaltar que para comunicar mais de um *data-server* e/ou *render-server*, é necessário o uso da biblioteca MPI (*Message Passing Interface*) [9] para o gerenciamento dos processos remotos e do tráfego de informação.

3. Padrões de comunicação

O padrão de comunicação do HeMoLab foi herdado do ParaView. Neste padrão, ações no cliente provocam a execução de algum tipo de processamento em um ou mais servidores (e.g., leitura ou escrita de dados, processamento de filtros, atualização de *widgets*, etc.). Os componentes mais importantes e suas responsabilidades na comunicação entre os processos servidores e o cliente são:

- ***Server Manager*** – é composto por classes especializadas responsáveis pela comunicação entre cliente e servidores (*proxies)*;
- ***Process Module*** – é responsável pelo gerenciamento e criação de objetos de maneira remota e por associar a cada um destes objetos identificadores globais únicos;
- ***Streams* cliente-servidor** – são utilizados para enviar mensagens a objetos remotos escritos na linguagem cliente-servidor;
- **Linguagem cliente-servidor** – é uma linguagem independente de plataforma utilizada para chamar métodos (enviando os parâmetros correspondentes) em objetos remotos utilizando o seu Id;
- **Interpretes cliente-servidor** – são utilizados para codificar/decodificar os *streams* enviados entre objetos em um formato independente de plataforma (o que permite a comunicação entre processos sendo executados em diferentes sistemas operacionais).

A associação entre componentes de interface, *proxies* e filtros VTK é feita utilizando arquivos XML (*Extensible Markup Language*) [10] especializados, os quais sejam:

- **XML Cliente** – permite inserção de novos componentes na interface do sistema de forma rápida associando-os a *proxies* especializados responsáveis por enviar informações aos diferentes filtros no servidor;
- **XML Servidor** – estabelece a relação entre *proxies* (responsáveis pela comunicação) e classes VTK no servidor permitindo a comunicação entre cliente e servidor. Estes *proxies* podem ser utilizados para enviar ou retornar dados aos servidores (*render-servers* e *data-servers*) através de propriedades XML.

A inserção de novos leitores, escritores, filtros ou *widgets* no HeMoLab é feita através dos arquivos XML Cliente e Servidor. No XML Cliente cria-se um novo módulo ao qual se atribui um nome único, e, por fim se constrói toda sua interface. Enquanto no XML Servidor, é implementado um novo *proxy*, que o vincula ao módulo cliente correspondente, o qual deve conter informações sobre a classe do lado servidor correspondente e as propriedades que serão utilizadas para enviar e obter informações do objeto então instanciado.

As classes que têm o papel mais importante na comunicação distribuída são `vtkProcessModule` e `vtkClientServerInterpreter`. Esta comunicação baseia-se no envio de mensagens (utilizando *streams* cliente-servidor) entre diferentes processos, permitindo o envio de dados de tipos básicos (e.g., *floats, integers, chars,* etc.), utilizando esta informação como parâmetro nos métodos das classes correspondentes. A classe `vtkProcessModule` cumpre um papel fundamental nesta

comunicação, atribuindo Ids globais únicos aos diferentes objetos em cada processo. Esta classe é uma instância *Singleton* (por processo) [11], a qual se mantém sincronizada entre os servidores e permite a chamada de métodos nos diferentes objetos e nos diferentes processos (sejam clientes ou servidores).

Instâncias da classe `vtkClientServerInterpreter` são utilizadas para traduzir as mensagens escritas na linguagem cliente-servidor enviadas/recebidas em chamadas a métodos nos objetos em diferentes processos, em diferentes computadores e até mesmo em diferentes sistemas operacionais, permitindo a sua comunicação. Desta forma, uma vez que as mensagens são interpretadas por esta classe, os Ids são traduzidos em endereços de memória, o que permite chamar os métodos nos objetos locais correspondentes. Por sua vez, os *proxies* armazenam os Ids dos objetos sob a sua responsabilidade, sendo responsáveis por uma interface simples e transparente para a comunicação cliente-servidor (a qual pode ser relativamente complexa). Na Figura 2 é possível visualizar o esquema de comunicação entre os servidores e o módulo cliente, através da utilização do componente *Process Module*.

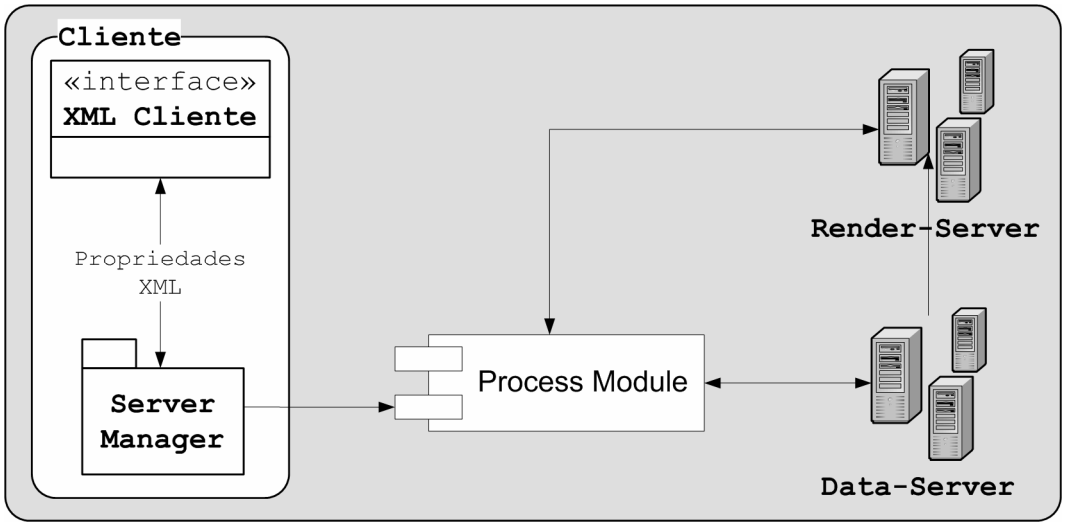

Figura 2. Padrão de Comunicação Distribuída do HeMoLab

3.1. Comunicação Distribuída utilizando *STREAMS*

O envio de mensagens entre processos é responsabilidade dos *proxies*. Ao montar uma *stream*, o *proxy* deve primeiramente configurar a ação a ser executada (e.g., criação de uma nova instância de alguma classe, remoção de algum objeto da memória ou invocação de um método). Logo em seguida é enviado o Id do objeto no qual o método será invocado, seguido do nome do método e dos devidos parâmetros. Para enviar a *stream*, o *proxy* utiliza uma referência do *singleton Process Module* indicando a qual processo esta *stream* será enviada.

Segue um exemplo (extraído do *proxy* do leitor de volumes DICOM do módulo de processamento de imagens do HeMoLab) de como se poderia enviar o diretório do volume de imagens a ser carregado para o objeto localizado no *data-server*, responsável pela efetiva leitura dos dados.

```
// O Process Module irá enviar o stream.
vtkProcessModule* pm = vtkProcessModule::GetProcessModule();
vtkClientServerStream stream;
char* DirectoryName = "/home/data/dicom/";

// O Id é utilizado para identificar o objeto que receberá
// a mensagem.
stream << vtkClientServerStream::Invoke <<  this->GetID(0)
       << "SetDirectoryName" << DirectoryName
       << vtkClientServerStream::End;
// O stream é enviado.
pm->SendStream(vtkProcessModule::DATA_SERVER, stream);
```

3.2. Comunicação Distribuída - Configuração via XML

A inserção de ferramentas no HeMoLab com baixo acoplamento torna-se uma tarefa relativamente simples com a utilização de arquivos de configuração no formato XML.

Toda a comunicação via XML é baseada em *proxies*, logo, utiliza diretamente o componente *Server Manager (*ou Gerenciador de Servidores) cuja responsabilidade é de viabilizar a comunicação entre o *render-server*, o *data-server* e o lado cliente, onde reside. Também é responsável por manter uma cópia do estado dos objetos dos servidores por meio do uso de propriedades. Nele existem referências a *proxies* (objetos locais para o cliente) que se comunicam com os objetos VTK (instanciados remotamente) nos servidores, fornecendo também uma interface para criação e gerenciamento de objetos naqueles processos possivelmente em paralelo. O *Server Manager* possui os seguintes componentes:

- ***Proxies*** – `vtkSMProxy` e suas subclasses fornecem uma interface local para se comunicar com os objetos VTK, que residem no(s) servidor(es). As suas responsabilidades são as seguintes: Criar os objetos VTK no servidor; Manter referências aos objetos VTK no servidor usando os Ids Cliente Servidor (CSID)[1]; Manter uma cópia do estado dos objetos que se encontram no lado servidor;
- **Propriedades** – `vtkSMProperty` e suas subclasses fornecem o acesso do cliente à interface (API - *Application Programming Interface*) dos objetos no lado servidor representados pelos *proxies*;
- **Domínios** – `vtkSMDomain` e suas subclasses representam uma coleção/intervalo de possíveis valores que as propriedades podem assumir.

Na prática, implementa-se no XML Cliente toda a interface vinculando seus componentes (e.g., combo boxes, entradas de texto, barras de rolagem, etc.) a propriedades existentes no XML Servidor. Antes do processamento dos objetos VTK nos servidores, as propriedades do XML Servidor são preenchidas a partir dos componentes de interface e executam métodos no objeto a este *proxy* (dono da propriedade) vinculado, passando os valores das propriedades como parâmetros. As propriedades podem ser utilizadas também para retornar para o cliente, dados de objetos VTK localizados nos servidores. A Figura 3 apresenta toda a estrutura de comunicação de um filtro genérico de imagens, utilizando inclusive um *widget* de visualização de volumes de imagens.

[1] O Id cliente servidor é um valor único em nível de sistema que permite identificar objetos no(s) servidor(es).

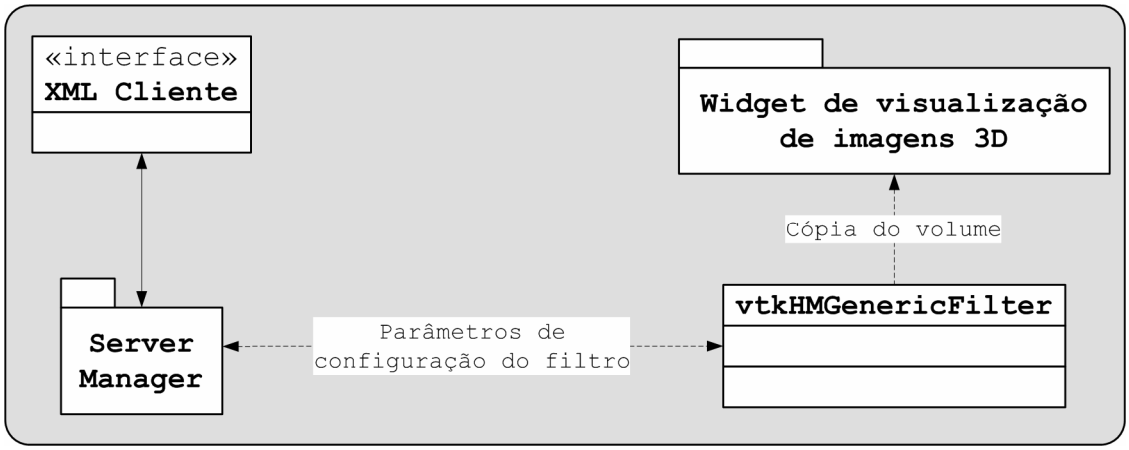

Figura 3. Estrutura do `GenericImageFilter`

4. Resultados

A forma mais utilizada para incorporar dados específicos de pacientes a modelos computacionais de sistemas médicos é utilizando imagens médicas como Tomografias Computadorizadas (TC), Ressonâncias Magnéticas (RM), Ultra-Sons (US), etc. Para isto, são necessárias ferramentas que permitam tratar estas imagens e extrair delas as informações geométricas que sejam relevantes para o tipo de estudo que se deseja realizar [12, 13, 14]. Neste sentido, vários pontos são relevantes:

- Leitura de dados em diferentes formatos;
- Remoção de ruído;
- Segmentação e identificação de regiões de interesse;
- Reconstrução das geometrias 3D presentes nas imagens.

Dando continuidade, a seguir são apresentadas as ferramentas implementadas no sistema HeMoLab, que, dentro dos padrões arquiteturais e de comunicação apresentados, se propõem a tratar cada uma das necessidades apontadas.

4.1. Leitura e Visualização de Imagens

Estes dados podem ser fornecidos em diversos formatos (e.g., jpg, bmp), sendo o mais comum o DICOM (*Digital Imaging and Communications in Medicine*). As imagens podem ser de diversos tipos, tais como: imagens 2D, seqüências temporais de imagens (na forma de filmes, e.g., US) ou imagens tridimensionais (dados volumétricos), sendo esta última a de maior interesse neste trabalho. Desta forma, as imagens 3D são interpretadas como conjuntos de *voxels* (i.e., elementos da imagem com comprimento, largura e profundidade, onde o volume pode ser interpretado como uma pilha de imagens 2D). Além dos dados correspondentes a imagem, o formato DICOM possui um cabeçalho com outras informações (e.g., nome do paciente, dados de escala e dimensões dos *voxels*, etc.).

A informação da imagem lida é armazenada em uma instância da classe `vtkImageData` do VTK. Por padrão, ao ler um volume de imagens no HeMoLab, o *render-server* cria na janela de renderização um cubo para representá-lo. A fim de oferecer ao usuário uma forma mais prática e interativa para visualizar o conteúdo de volumes de imagem, foi implementado um *widget* 3D (`ImageWorkspaceWidget`), o qual cria um ambiente de visualização de imagens 3D. Dentre suas funcionalidades, destacam-se seus três planos móveis alinhados aos eixos X, Y e Z capazes de percorrer todo o volume, refatiando-o e exibindo seu conteúdo a partir destes eixos. A associação

entre o leitor/filtro e o *widget* responsável pela sua visualização de seus dados é especificada utilizando os arquivos XML mencionados anteriormente.

Mesmo com o alto desempenho do *hardware* computacional da atualidade, aplicações médicas ainda apresentam baixo desempenho devido ao grande volume de informações com que lidam e ao custo de se processar cálculos sobre estas. No módulo de processamento de imagens, pode-se dar como exemplo a leitura e manipulação de tomografias digitais, que são constituídas por dezenas ou até mesmo centenas de imagens 2D somadas a suas informações de cabeçalho. Baseado nisso, foi criado o filtro `ExtractGrid`, que viabiliza a seleção de sub-volumes através de um cubo interativo.

4.2. Processamento de imagens

Tratando-se de imagens médicas, estas podem estar degradadas por diversos fatores (e.g., movimento do paciente, limitações nos aparelhos de aquisição, etc.). Tais características dificultam o processo de identificação das estruturas de interesse. Para pré-processar as imagens, alguns filtros de melhoramento e restauração amplamente utilizados foram incorporados no módulo de processamento de imagens do HeMoLab, quais sejam: suavizado Gaussiano**,** filtro da mediana 3D e difusão anisotrópica [15].

Uma vez que o volume foi pré-processado, a região de interesse deve ser selecionada. Este processo é chamado de segmentação de imagens [15, 16]. Foram incorporados no HeMoLab os seguintes algoritmos de segmentação:

- **Voxel Grow** – realiza a segmentação através do filtro de Crescimento de Regiões, onde as regiões crescem a partir de *voxels* escolhidos na imagem pelo próprio usuário (sementes) [17];
- **Dilate/ Erode** – aplica o filtro de Dilatação/ Erosão de imagens [15];
- **Open/Close 3D** – aplica o filtro Open/Close em volumes de imagens [15];
- **Topological Derivative** – realiza a segmentação da imagem através do filtro da Derivada Topológica [18].

Nas imagens estudadas, estes filtros foram suficientes para segmentar de maneira adequada a maioria das imagens. No entanto, algumas imagens (dependendo do nível do ruído, contraste, etc.) não foram segmentadas corretamente. Por este motivo outros filtros mais complexos (e.g., baseados em *Level Sets*) estão sendo incorporados na ferramenta, permitindo assim processar a totalidade das imagens.

4.3. Reconstrução de Geometrias

Uma vez selecionada a região de interesse dentro da imagem, a geometria (triangulação) que a representa é gerada. A partir desta primeira versão da geometria do vaso, filtros de suavização de malhas podem ser facilmente aplicados. No HeMoLab foi adotado um filtro de geração de contorno para a construção de superfícies 3D a partir de imagens tridimensionais.

Existem em outros módulos do HeMoLab, ferramentas que permitem o tratamento e a geração de modelos (e.g., suavização, edição, refinamento, geração de volume, etc.) a partir desta primeira versão da geometria dos vasos. Posteriormente, estes modelos podem ser acoplados a outros módulos responsáveis por simulações e, efetivamente, por gerar informações as quais irão auxiliar no processo de predição de resultados de intervenções cirúrgicas.

Com o desenvolvimento e incorporação do módulo de processamento de imagens ao HeMoLab, este sistema computacional é capaz de realizar a leitura de imagens médicas, tratá-las, extrair regiões de interesse e gerar malhas tridimensionais.

Assim, o uso de imagens médicas é a forma mais usual de incorporar informações específicas do paciente nos anteriormente citados modelos do Sistema Cardiovascular Humano.

Na Figura 4 pode ser acompanhado um caso de estudo desde a leitura de uma tomografia computadorizada até a geração da malha 3D, em que temos as seguintes etapas, respectivamente:

a. Leitura de uma tomografia computadorizada no formato DICOM;
b. Seleção de parte da tomografia;
c. Aplicação do filtro da difusão anisotrópica;
d. Segmentação da imagem 3D utilizando o filtro da Derivada Topológica;
e. Seleção da região segmentada com o filtro Voxel Grow;
f. Malha gerada com o filtro de contorno e suavizada com ferramentas de outros módulos do HeMoLab.

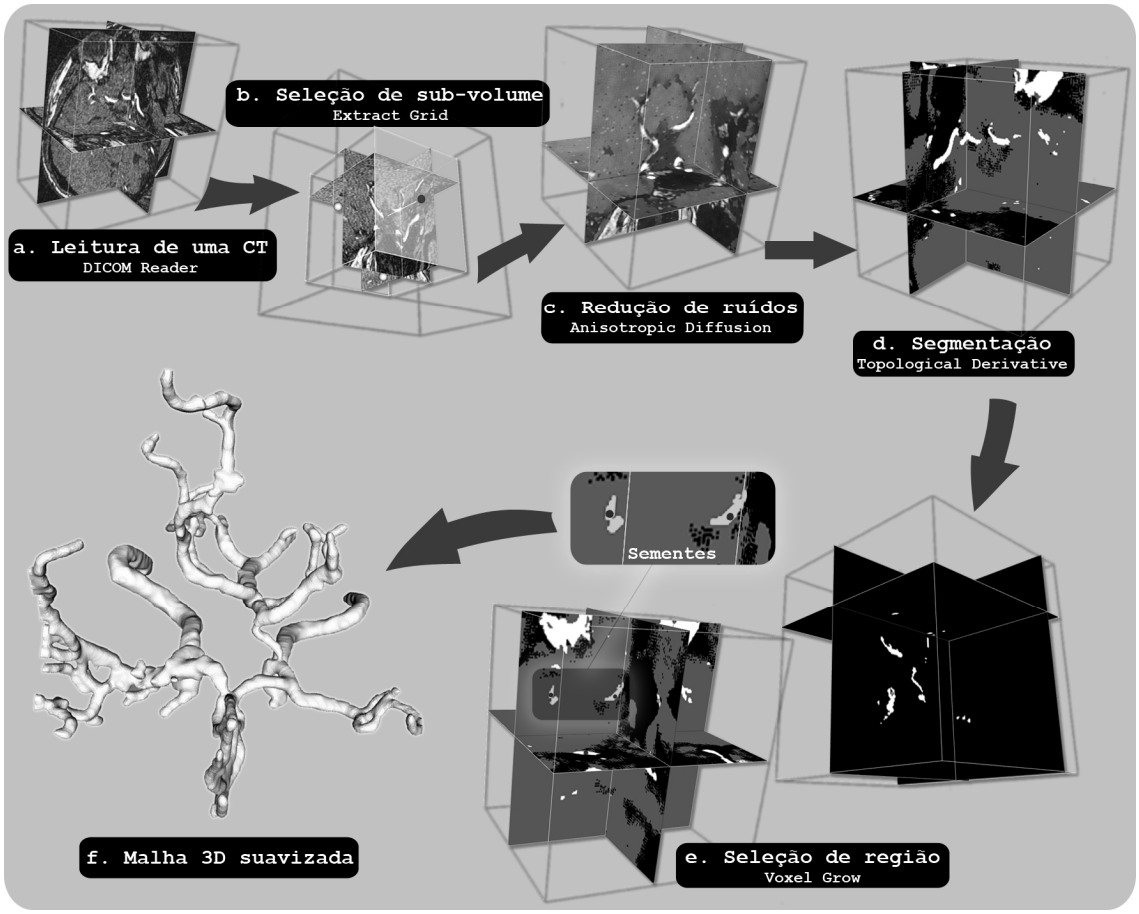

Figura 4. Leitura de Tomografia Computadorizada à geração da malha 3D

5. Conclusões

Com base no crescente aumento do número de incidências de doenças cardiovasculares e da complexidade das intervenções cirúrgicas nos últimos anos, surgiu a necessidade de utilizar técnicas e ferramentas que permitem predizer de maneira precisa e não invasiva resultados destes procedimentos cirúrgicos. Usufruindo do rápido avanço das técnicas de modelagem e simulação computacional, somadas ao poder do

processamento distribuído, o HeMoLab oferece uma completa suíte de ferramentas para o tratamento de doenças cardiovasculares.

Fazendo uso dos leitores de imagens, o HeMoLab é efetivamente capaz de realizar a leitura de imagens médicas e as inserir no *pipeline* da aplicação. O *widget* de visualização de imagens 3D apresenta-se eficaz e eficiente em sua tarefa de exibir de forma prática conteúdos de imagens volumétricas, seja para a exibição de imagens 3D lidas ou das entradas e saídas de filtros de imagem. Com os filtros de processamento de imagem é possível tratar e extrair regiões de interesse a partir de qualquer imagem previamente lida. Após a geração e tratamento da geometria dos vasos, podem-se gerar os modelos computacionais destes e então adicioná-los aos modelos simplificados do Sistema Cardiovascular Humano, a fim de realizar simulações que permitirão ao médico prever como o paciente reagiria a um procedimento cirúrgico.

A utilização do módulo de processamento de imagens é de vital importância para que, quando o HeMoLab for utilizado pela comunidade médica, forneça informações precisas sobre os pacientes, nas quais esta precisão implicará diretamente na qualidade dos resultados. Estas informações serão extremamente valiosas nos diagnósticos e conseqüentes tratamentos das doenças cardiovasculares.

6. Referências

[1] WHO World Health Organization. http://www.who.int/whosis/whostat2006/en/index.html. WHO - World Health Organization, 2002.

[2] S. Urquiza, P. J. Blanco, G. Lombera, M. J. Vénere, and R. A. Feijóo. Blood flow finite element solution through the carotid artery. In *Mecânica Computacional*, volume 22, Bahia Blanca - Argentina, 2003.

[3] Larrabide, I., Feijo, R. A, HeMoLab: Laboratório de Modelagem em Hemodinâmica; Relatório Técnico 13/2006, LNCC, Brasil, 2006.

[4] L. Formaggia, J. F. Gerbeau, F. Nobile, and A. Quarteroni. On the coupling of 3d and 1d navier-stokes equations for flow problems in compliant vessels. *Comp. Meth.App. Mech & Eng.*, 191:561-582, 2001.

[5] A. Quarteroni, M. Tuveri, and A. Veneziani. Computational vascular fluid dynamics:problems, models and methods. Comp. Vis. Science, 2:163–197, 2000.

[6] Kitware. ParaView - Kitware Inc. http://www.paraview.org.

[7] Kitware. Vtk: The visualization toolkit - kitware inc. http://www.vtk.org.

[8] OpenGL. OpenGL. http://opengl.org.

[9] MPI. http://www-unix.mcs.anl.gov/mpi/

[10] http://www.w3.org/TR/REC-xml/

[11] E. Gamma, R. Helm, and R. Johnson. *Design Patterns. Elements Of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series. Addison-Wesley, 1995. Gam E 95:1 1.Ex.

[12] A. S. Frangakis and R. Hegerl. Noise reduction in electron tomographic reconstructions using nonlinear anisotropic diffusion. J. Struct. Biol., 1(135):239–250, 2001.

[13] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Anal. Machine Intell.*, 12(7):629–639, 1990.

[14] J. Suri, S. Singh, and K. Setarehdan, editors. Advanced algorithmic approaches to medical image segmentation: state-of-the-art applications in cardiology, neurology, mammography and pathology. Springer-Verlag, 2001.

[15] C. R. Gonzalez and R. E. Woods. Digital Image Processing - Second Edition. Prentice Hall, 2001.

[16] Y. Zhang, M. Brady, and S. Smith. Segmentation of brain mr images through a hidden markov random field model and the expectation-maximization algorithm. *IEEE Transactions on Medical Imaging,* 20(1):45–57, 2001.

[17] Larrabide, I., Fiorentini, S., and Vénere, M. J. 2003. Voxel grow: A region growing segmentation technique. In International Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Applications (CSITeA03).

[18] Larrabide, I., Feijóo, R. A., Novotny, A. A., Taroco, E., and Masmoudi, M. 2005. An image segmentation method based on a discrete version of the topological derivative. In World Congress Structural and Multidisciplinary Optimization 6, Rio de Janeiro. International Society for Structural and Multidisciplinary Optimization.