Visualização Interativa em Tempo Real de Dados Médicos na Web

Thiago Franco Moraes
Paulo Henrique Junqueira Amorim
Jorge Vicente Lopes da Silva
Centro de Tecnologia da Informação Renato Archer
Campinas-SP, Brasil, 13069-901

Hélio Pedrini
Instituto de Computação
Universidade Estadual de Campinas
Campinas-SP, Brasil, 13083-852

***Resumo*—Em várias áreas de aplicação, tais como medicina, sensoriamento remoto, metereologia e biologia, há uma demanda pela visualização de grandes volumes de dados. Na forma tradicional de visualização científica, os dados são manipulados e exibidos localmente na máquina-cliente, o que pode requerer recursos computacionalmente custosos em termos de espaço de armazenamento e poder de processamento. Este trabalho apresenta um sistema para visualização 2D e 3D remota de dados médicos na Web. A partir de requisições enviadas a um servidor, usuários são capazes de interagir com volumes de dados de forma rápida e simples no próprio ambiente de navegação Web.**

***Keywords*-visualização remota; dados médicos; visualização na Web.**

I. Introdução

A visualização de dados é uma atividade com aplicações em diversos domínios de conhecimento, tais como medicina, biologia, sensoriamento remoto, meteorologia, entre outros.

O aumento crescente no volume de dados dificulta o processo de visualização tradicional realizado localmente em uma máquina-cliente, uma vez que depende de recursos computacionalmente caros para manipular e exibir os volumes de dados. Dessa forma, a máquina-cliente deve apresentar capacidade suficiente de processamento e armazenamento.

Por outro lado, a visualização remota de dados [1], [2], [3], [4] permite que operações custosas sejam realizadas em um servidor com alta capacidade de processamento e armazenamento, com recursos para compartilhar a colaboração entre múltiplos usuários, com mecanismos de segurança e com maior facilidade no controle de versão das aplicações.

A elevada quantidade de dados produzida na área médica requer o uso de técnicas eficientes de análise de imagens, as quais normalmente envolvem intervenção dos especialistas para permitir maior controle da aplicação. Exemplos de aplicações médicas interativas são segmentação de imagens, simulação de cirurgias, realidade virtual e aumentada, telemedicina, diagnóstico auxiliado por imagens, entre outras.

Este trabalho apresenta um sistema para visualização remota de dados médicos na Web. A partir de requisições enviadas a um servidor, usuários são capazes de manipular e visualizar grandes volumes de dados de forma rápida e simples no próprio ambiente de navegação Web. Vários benefícios são oferecidos aos usuários a partir do sistema proposto. A renderização por meio de navegadores dispensa a instalação de aplicativos especiais na máquina-cliente. A solução proposta utiliza apenas pacotes livres e gratuitos. O servidor é o responsável pela execução das operações requisitadas pelos usuários, o que prescinde a máquina-cliente de requisitos caros em termos de memória e processamento. Além disso, os resultados da visualização tornam-se disponíveis a partir de qualquer dispositivo com acesso à Internet.

O texto está organizado da seguinte forma. A seção II apresenta conceitos e trabalhos relacionados ao tema sob investigação. A seção III descreve a metodologia de desenvolvimento de sistema de visualização remota de dados médicos na Web. Os resultados obtidos a partir da aplicação da metodologia são apresentados e discutidos na seção IV. Finalmente, a seção V conclui o trabalho e apresenta sugestões para trabalhos futuros.

II. Conceitos e Trabalhos Relacionados

Imagens médicas são tipicamente adquiridas por meio de equipamentos de tomografia computadorizada e de ressonância magnética, em que seções transversais em níveis de cinza da região anâtomica de interesse são capturadas e armazenadas.

Várias atividades de análise de imagens podem ser realizadas pelos especialistas médicos a partir de uma ferramenta de diagnóstico médico baseado em imagens, tais como realce de áreas de interesse, segmentação, fusão de imagens, reconstrução tridimensional e visualização.

Com o avanço dos sistemas computacionais e a popularização do uso da Internet, o usuário passou a ter acesso a informações multímidia, tais como imagens, vídeos, áudio e textos, de forma rápida e simples.

Nesse sentido, a Realidade Virtual e Aumentada [5], [6], [7] provê um conjunto de recursos para o desenvolvimento de aplicações que normalmente requerem avançada interação, permitindo a simulação de ambientes com elevado nível de detalhes a partir de dados complexos [8].

A linguagem VRML (*Virtual Reality Modeling Language*) [9] foi proposta para a modelagem de Realidade

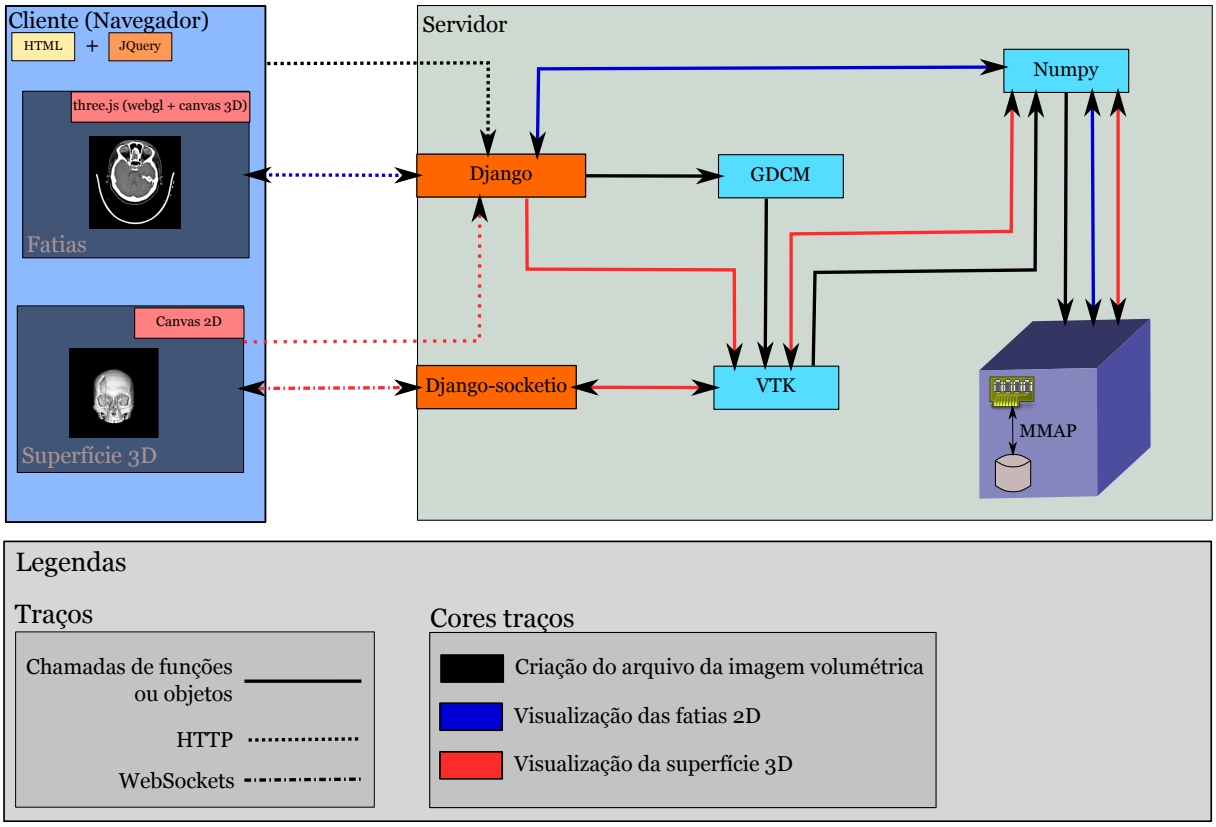

Figura 1: Diagrama ilustrando os principais componentes da metodologia desenvolvida.

Virtual em ambientes Web, permitindo que usuários conectados à Internet possam visualizar interativamente volumes de dados e transmitir arquivos.

O trabalho de Mahmoudi et al. [10] faz uso do padrão VRML para a exibição de imagens médicas tridimensionais, em conjunto com Ajax e Javascript para a construção da interface gráfica. Para realizar o processamento e a reconstrução tridimensional das imagens, por parte do servidor, as bibliotecas ITK [11] e VTK [12], [13] são utilizadas. Embora a maior carga de processamento fique na camada do servidor, os arquivos VRML precisam ser carregados na camada do cliente para que possam ser visualizados, gerando picos de tráfego na rede e requerendo grande capacidade de memória na camada do cliente. Além disso, o navegador deve possuir instalado um *plugin* para visualização de arquivos desse tipo, deixando a aplicação dependente de instalações de terceiros.

Settapat et al. [14] propõem um *framework* para a visualização de imagens médicas via Web utilizando o padrão X3D [15], que é baseado em VRML. Quando comparado com VRML, o padrão X3D prevê a escrita de dados em formato binário, assim diminuindo o tamanho dos arquivos. Além disso, ele suporta vários outros formatos de arquivos para escrita, por exemplo, o XML [16]. Embora o padrão X3D traga várias melhorias quando comparado com VRML, ele ainda necessita de *plugins* de terceiros instalados no navegador do cliente para que seja possível visualizar os objetos renderizados. Behr e Alexa [17] também apresentam um sistema de renderização de volumes que integra VRML e X3D.

Marion e Jomier [4] descrevem a arquitetura de um sistema colaborativo em tempo real que combina WebGL [18], [19] e WebSocket [20]. Jomier et al. [21] apresentam uma solução baseada em ParaViewWeb [22] para computação em nuvens para armazenamento de bases de dados e visualização tridimensional. Congote et al.[23] descrevem um sistema de renderização que utiliza WebGL para visualizar dados médicos e de meteorologia.

Poliakov et al. [24] descrevem um sistema de visualização interativa baseada na Web para análise de dados na área de neurociências. A máquina-cliente utiliza linguagem de programação Java e CGI (*Common Gateway Interface*) para realizar as requisições ao servidor.

III. Metodologia

Utilizando-se recursos modernos disponíveis nos navegadores (*browsers*) atuais, tais como *WebGL* [18] e *WebSocket* [20], em conjunto com a arquitetura cliente-servidor, tornou-se possível desenvolver um visualizador de imagens que não demanda recursos computacionais avançados, uma

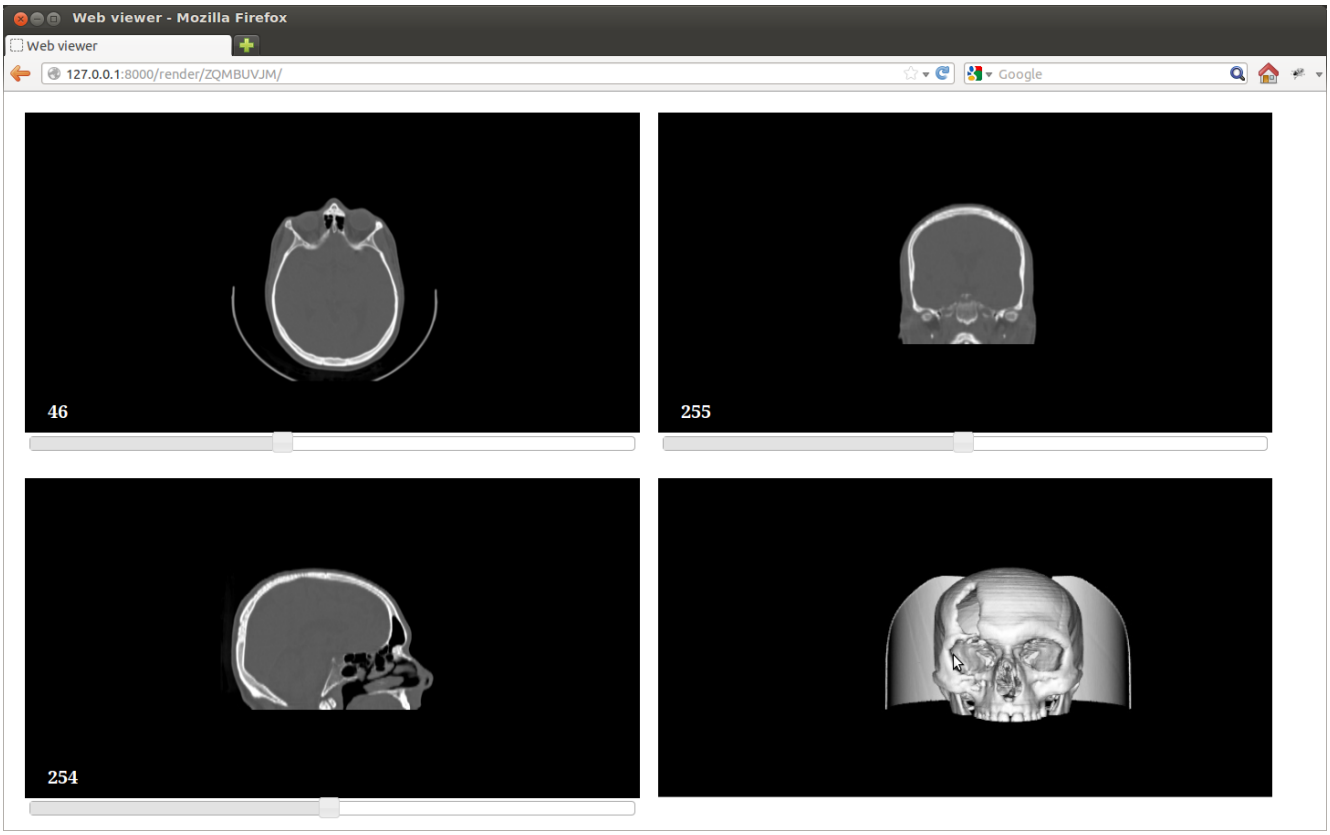

Figura 2: Interface do sistema utilizada para a visualização das fatias nas orientações axial, coronal e sagital e para a visualização do volume.

vez que o processamento pode ser realizado no servidor, cabendo ao cliente a interação com o ambiente e a visualização dos resultados.

A metodologia desenvolvida neste trabalho está ilustrada no diagrama da Figura 1. As principais etapas envolvidas no processo são:

- criação do arquivo da imagem volumétrica: as imagens de entrada são armazenadas em memória secundária usando um formato de fácil acesso.
- visualização das fatias bidimensionais: as imagens de interesse do usuário são transmitidas do servidor ao navegador do cliente para sua visualização.
- visualização da superfície tridimensional: utiliza os recursos de *websockets* e de renderização em memória para permitir a visualização remota.

O sistema proposto neste trabalho foi desenvolvido utilizando-se a linguagem de programação Python em conjunto com o *framework Web* Django [25] na camada do servidor e *JavaScript* na camada do cliente.

A. Criação do arquivo da imagem volumétrica

O fluxo de trabalho da metodologia inicia-se com o envio de imagens médicas no formato DICOM [26] do cliente para servidor. No servidor, a biblioteca GDCM [27] é utilizada para reconstruir as imagens. A biblioteca GDCM foi escolhida por ser de código aberto e pela capacidade de interpretar imagens DICOM originados dos variados tipos de equipamentos de tomografia computadorizada e ressonância magnética.

Após a reconstrução das imagens, o volume resultante é convertido em uma matriz volumétrica utilizando a biblioteca numérica *Numpy* [28] em conjunto com a biblioteca VTK (Visualization Toolkit) [12], [13]. A matriz volumétrica é então gravada em um arquivo na memória secundária.

O arquivo é mapeado em memória via MMAP [29] utilizando o *Numpy*, tal que apenas as porções dos dados necessárias ao processamento são mantidas em memória principal e não todo o volume.

Após a preparação dos dados, tanto bidimensionais quanto tridimensionais, para a visualização, todo o processamento das etapas posteriores é realizado no servidor, bastando ao cliente apenas requisitar dados ao servidor e interpretá-los.

B. Visualização das fatias bidimensionais

Esta etapa visa permitir a visualização das fatias nas orientações axial, sagital e coronal. Ela inicia-se com a requisição, por parte do cliente, utilizando o protocolo

HTTP (*Hypertext Transfer Protocol*) [30], de uma dada fatia segundo orientação de interesse.

Antes de atender a essa requisição, o servidor deve recuperar a fatia de interesse no arquivo contendo a matriz volumétrica. A recuperação da fatia é feita por meio de acesso do arquivo com *Numpy* e *MMAP*.

De posse da imagem, o cliente renderiza-a na tela do navegador. A renderização é feita mapeando a imagem em um plano, tal como uma textura. A biblioteca *JavaScript three.js* [31] é utilizada para a renderização, visto ser possível utilizar *WebGL* ou *canvas 2D* de modo transparente, sendo que *WebGL* não é suportado em todos os navegadores, principalmente aqueles presentes em dispositivos móveis.

C. Visualização da superfície tridimensional

Esta etapa inicia-se com a requisição do cliente, via protocolo HTTP. Para atender a essa requisição, o servidor necessita extrair a malha de triângulos da matriz volumétrica, que é realizada pelo uso do método de cubos marchantes (*marching cubes*) [32], disponível na biblioteca VTK. Em seguida, a visualização é realizada no servidor utilizando-se a técnica de renderização *offscreen* [33].

O protocolo *WebSocket* permite a comunicação bidirecional entre cliente e servidor. Ao contrário do protocolo HTTP, a comunicação entre ambos se mantêm ativa, o que facilita a transmissão e o desenvolvimento de conteúdo em tempo real, característica necessária neste trabalho.

A implementação *WebSocket* utilizada pelo servidor é a implementada pela biblioteca *gevent-socketio* [34], em conjunto com a biblioteca *django-socketio* [35] para permitir a comunicação entre a biblioteca *gevent-socketio* e o *Django*. O protocolo *WebSocket* é implementado diretamente no navegador do cliente, o que elimina a necessidade da instalação de um pacote ou *plugin* específico no cliente.

A visualização da superfície 3D pelo cliente é realizada pelo envio de mensagens de requisição de renderização ao servidor via *Websockets*. As requisições são feitas utilizando um protocolo próprio, em que são transmitidos comandos de manipulação de câmera ao servidor, podendo então ser realizadas transformações de rotação e escala e, assim, interagir com a malha de triângulos.

O servidor recebe esses comandos, aplica as mudanças na câmera e realiza a renderização *offscreen*, que é armazenada em uma imagem bidimensional no formato JPEG [36]. O servidor, então, responde às requisições enviando a imagem resultante ao canvas do navegador do cliente.

IV. Resultados

Os resultados apresentados nesta seção foram obtidos com um servidor cuja configuração é constituída por 2 processadores Intel Xeon de 2,80 GHz, 48GB de memória principal, placa de vídeo NVidia Quadro 400 com 2GB de memória e sistema operacional Linux Ubuntu 12.04, em uma rede local *gigabit*.

O cliente utilizou um navegador Firefox em um sistema *desktop* para ter acesso aos recursos de visualização de fatias e do volume com grande grau de interatividade. A interface do sistema de visualização remota para as fatias 2D e a imagem reconstruída 3D é ilustrada na Figura 2.

A metodologia se mostrou muito flexível por permitir utilizar o sistema não apenas em computadores do tipo *desktop*, mas também em dispositivos móveis como *smartphones* (iPhone e Android) e *tablets*, conforme demonstrado nas Figuras 3, 4 e 5, respectivamente.

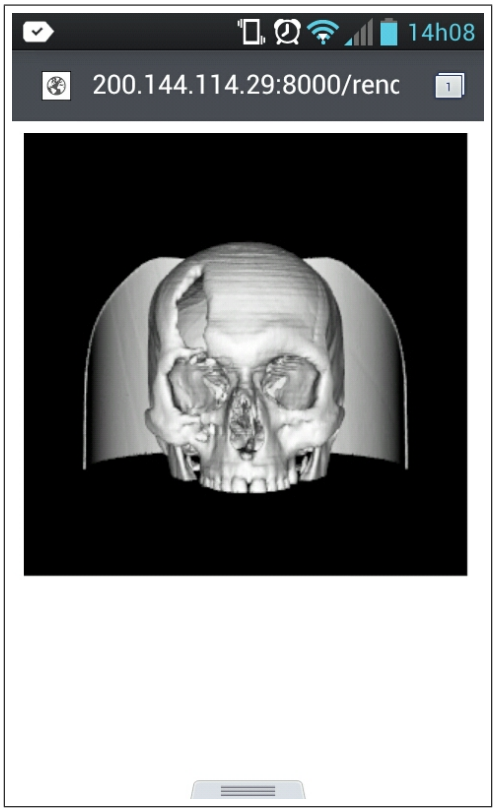

Figura 3: Resultado da visualização de volume em um *smartphone* Android versão 4.0.

O sistema desenvolvido provê vários benefícios aos usuários da área médica interessados na visualização remota de dados bidimensionais e tridimensionais. A visualização por meio de navegadores é simples e não requer a instalação de aplicativos especiais na máquina-cliente. O sistema é baseado apenas em pacotes livres e gratuitos, o que reduziu os custos de desenvolvimento e manutenção do sistema. A configuração da máquina-cliente não necessita de requisitos caros em termos de memória e processamento, uma vez que o servidor é o responsável pela execução das operações requisitadas pelos usuários. Finalmente, a visualização pode ser realizada não apenas em computadores do tipo *desktop*, mas também em dispositivos móveis.

O desempenho do sistema foi avaliado por meio de

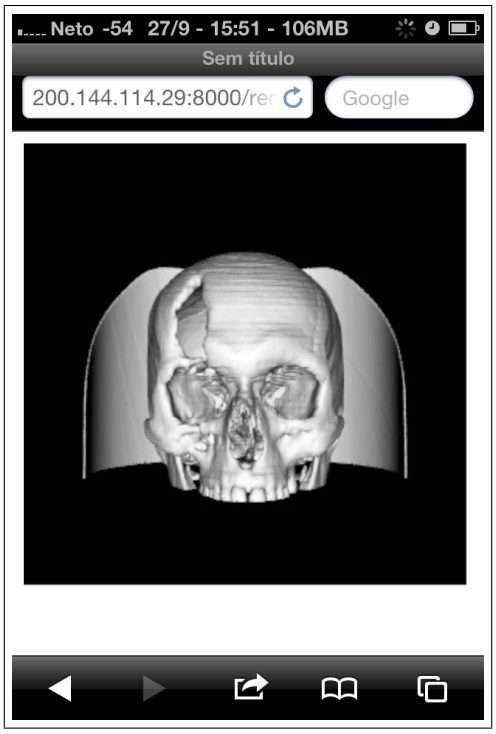

Figura 4: Resultado da visualização de volume em um *smartphone* iPhone.

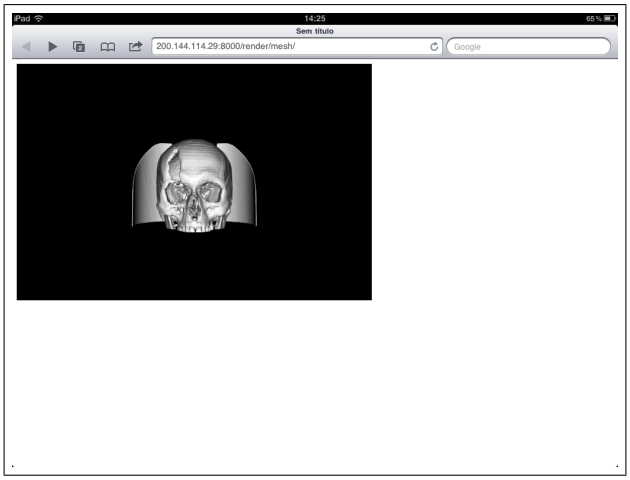

Figura 5: Resultado da visualização de volume em um *tablet* iPad.

experimentos em que o cliente, ao visualizar modelos de malhas de triângulos, envia 1000 comandos de manipulação de câmera ao servidor. Durante a renderização, medições da taxa de quadros por segundo são realizadas, repetidas três vezes para cada modelo. Os resultados apresentados nesta seção referem-se a cinco modelos diferentes de malhas de triângulos. A Tabela I mostra o número de vértices e faces de cada modelo, bem como a respectiva taxa de quadros por segundo.

Tabela I: Número de vértices e arestas dos modelos e taxa de quadros por segundo para os cinco modelos de malhas de triângulos utilizados nos experimentos.

Modelos	Vértices	Faces	Taxa de quadros por segundo
01	3162	6214	41,38
02	534523	1067186	30,45
03	732880	1466980	33,27
04	2008389	3913155	31,79
05	2449418	4903834	21,82

Pode-se observar na Tabela I que a taxa de quadros por segundo é inversamente proporcional ao número de vértices e faces dos modelos. A renderização de todos os modelos gerou taxas acima de 16 quadros por segundo, sendo valores considerados suficientes para se obter a sensação de continuidade visual e de movimento [37].

V. Conclusões e Trabalhos Futuros

Este trabalho descreveu um sistema para visualização remota de dados médicos na Web. A solução apresentada requer, na camada do cliente, apenas um navegador moderno com suporte à linguagem HTML5, WebGL e WebSocket, sem a necessidade de aplicativos ou *plugins* adicionais, como ocorre no caso de VRML e X3D.

O sistema de visualização remota concentra a execução das operações mais complexas no servidor, o qual transmite os resultados da visualização aos usuários conforme demanda. Essa centralização do processamento facilita o uso e a manutenção dos programas.

A visualização realizada diretamente no navegador do cliente torna-se simples aos usuários, dispensando o uso de recursos avançados na máquina-cliente.

Algumas propostas de trabalhos futuros incluem a utilização de estratégias mais compactas na transmissão das imagens, o uso de diferentes níveis de detalhe para facilitar a manipulação de malhas muito densas, a renderização distribuída pelo servidor e o emprego de representações *out-of-core* para malhas de triângulos.

Agradecimentos

Os autores são gratos ao Ministério da Saúde, à FAPESP, ao CNPq e à CAPES pelo apoio financeiro. Também agradecem a valiosa colaboração de Fábio de Souza Azevedo no processo de medição de desempenho do sistema.

Referências

[1] F. Evesque, S. Gerlach, and R. Hersch, "Building 3D Anatomical Scenes on the Web," *Journal of Visualization and Computer Animation*, vol. 13, no. 1, pp. 43–52, 2002.

[2] A. Wessels, M. Purvis, J. Jackson, and S. Rahman, "Remote Data Visualization through WebSockets," in *Eighth International Conference on Information Technology: New Generations*, Apr. 2011, pp. 1050–1051.

[3] B. Blazona and Z. Mihajlovic, "Visualization Service Based on Web Services," *Journal of Commputing and Information Technology*, vol. 4, pp. 339–345, 2007.

[4] C. Marion and J. Jomier, "Real-Time Collaborative Scientific WebGL Visualization with WebSocket," in *17th International Conference on 3D Web Technology*, Los Angeles, CA, USA, 2012, pp. 47–50.

[5] H. Rheingold, *Virtual Reality*, ser. A Touchstone Book Series. Simon & Schuster, 1992.

[6] O. Bimber and R. Raskar, *Spatial Augmented Reality: Merging Real and Virtual Worlds*. A.K. Peters, Ltd., Jul. 2005.

[7] J. Vince, *Introduction to Virtual Reality*. Springer, 2004.

[8] M. Kera, H. Pedrini, and F. Nunes, "Detecção de Colisões em Ambientes Virtuais para Treinamento Médico," in *Workshop de Realidade Virtual e Aumentada*, Nov. 2007.

[9] VRML, "Virtual Reality Modeling Language," 2012, http://www.vrml.org/.

[10] S. E. Mahmoudi, A. Akhondi-Asl, R. Rahmani, S. Faghih-Roohi, V. Taimouri, A. Sabouri, and H. Soltanian-Zadeh, "Web-based Interactive 2D/3D Medical Image Processing and Visualization Software," *Computer Methods and Programs in Biomedicine*, vol. 98, no. 2, pp. 172–182, 2010.

[11] ITK, "Insight Segmentation and Registration Toolkit," 2012, http://www.itk.org/.

[12] W. Schroeder, K. Martin, K. W. Martin, and B. Lorensen, *The Visualization Toolkit*. Prentice Hall PTR, 2004.

[13] VTK, "Visualization Toolkit," 2012, http://www.vtk.org/.

[14] S. Settapat, T. Achalakul, and M. Ohkura, "Web-based 3D Visualization and Interaction of Medical Data using Web3D," in *SICE Annual Conference 2010*, Aug. 2010, pp. 2986–2991.

[15] Web3D, "Consortium - Open Standards for Real-Time 3D Communication," 2012, http://www.web3d.org/realtime-3d/.

[16] XML, "Extensible Markup Language," 2012, http://www.w3.org/XML/.

[17] J. Behr and M. Alexa, "Volume Visualization in VRML," in *Sixth International Conference on 3D Web Technology*, Paderbon, Germany, 2001, pp. 23–27.

[18] WebGL, "OpenGL ES 2.0 for the Web," 2012, http://www.khronos.org/webgl/.

[19] A. Anyuru, *WebGL Programming: Developing 3D Graphics for the Web*. John Wiley & Sons, Ltd., 2012.

[20] I. Fette and A. Melnikov, "The WebSocket Protocol," RFC 6455 (Proposed Standard), Internet Engineering Task Force, Dec. 2011.

[21] J. Jomier, S. Jourdain, U. Ayachit, and C. Marion, "Remote Visualization of Large Datasets with MIDAS and ParaViewWeb," in *16th International Conference on 3D Web Technology*, Paris, France, 2011, pp. 147–150.

[22] ParaViewWeb, "Open Source Scientific Visualization," 2012, http://paraviewweb.kitware.com/.

[23] J. Congote, A. Segura, L. Kabongo, A. Moreno, J. Posada, and O. Ruiz, "Interactive Visualization of Volumetric Data with WebGL in Real-Time," in *16th International Conference on 3D Web Technology*, Paris, France, 2011, pp. 137–146.

[24] A. V. Poliakov, E. Albright, K. P. Hinshaw, D. P. Corina, G. Ojemann, R. F. Martin, and J. F. Brinkley, "Server-based Approach to Web Visualization of Integrated Three-Dimensional Brain Imaging Data," *Journal of the American Medical Informatics Association*, vol. 12, no. 2, pp. 140–151, 2005.

[25] Django, "Python Web Framework," 2012, https://www.djangoproject.com/.

[26] DICOM, "Digital Imaging and Communications in Medicine," 2012, http://medical.nema.org/.

[27] GDCM, "Grassroots DICOM Library," 2012, http://sourceforge.net/projects/gdcm/.

[28] T. E. Oliphant, "Python for Scientific Computing," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 10–20, 2007.

[29] MMAP, "MMAP," 2012, http://www.kernel.org/doc/man-pages/online/pages/man2/mmap.2.html.

[30] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616 (Draft Standard), Internet Engineering Task Force, Jun. 1999, http://www.ietf.org/rfc/rfc2616.txt.

[31] R. Cabello, "three.js - JavaScript 3D library," 2012, http://mrdoob.github.com/three.js/.

[32] W. E. Lorensen and H. E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," in *14th Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 1987, pp. 163–169.

[33] B. Paul, "OpenGL/Mesa Off-Screen Rendering," in *ACM SIGGRAPH Conference*, Los Angeles, CA, USA, Aug. 1997.

[34] A. Bourget, "gevent-socketio," 2012, https://github.com/abourget/gevent-socketio.

[35] S. McDonald, "django-socketio," 2012, https://github.com/stephenmcd/django-socketio.

[36] JPEG, "Joint Photographic Experts Group," 2012, http://www.jpeg.org/index.html.

[37] P. Read and M.-P. Meyer, *Restoration of Motion Picture Film*, ser. Series in Conservation and Museology. Butterworth-Heinemann, Sep. 2000.