colloquium exactarum

HVRview: UMA FERRAMENTA PARA RENDERING VOLUMÉTRICO HÍBRIDO

HVRview: A TOOL FOR HYBRID VOLUME RENDERING

Rafael Silva Santos, Danilo Medeiros Eler

Universidade Estadual Paulista – UNESP, Faculdade de Ciências e Tecnologia, Departamento de Matemática e Computação, Presidente Prudente, SP.
E-mail: rafael.silva.sts@gmail.com, daniloeler@fct.unesp.br

RESUMO - Ao longo dos anos, diferentes abordagens para visualização de volumes foram propostas na literatura. Neste trabalho, apresentamos uma nova ferramenta de visualização volumétrica, o *Hybrid Volume Rendering Visualizer* (HVRview). Em contraste às demais ferramentas, o HVRview incorpora uma abordagem de *Hybrid Volume Rendering* (HVR) – a técnica *Volume on Surface* (VoS) – com o objetivo de acelerar o processo de *rendering*. A ferramenta HVRview implementa uma versão sequencial em CPU e uma paralela em GPU da abordagem VoS. Em alguns testes, a execução paralela propiciou um *speedup* superior a 24 vezes.
Palavras-chave: *Rendering* Volumétrico Híbrido, *Volume on Surface*, Visualização Volumétrica.

ABSTRACT - Over the years, different tools for Volume Visualization have been proposed in the literature. In this paper, we present a new tool, the Hybrid Volume Rendering Visualizer (HVRview). In contrast to other tools, HVRview implements an approach of Hybrid Volume Rendering (HVR) – the Volume on Surface technique (VoS) – aiming to accelerate the rendering process. The HVRview tool implements a sequential version on CPU and a parallel version on GPU of the VoS approach. In some tests, the execution on GPU provided a speedup higher than 24 times.
Keywords: Hybrid Volume Rendering, Volume on Surface, Volume Visualization.

Recebido em: 16/08/2016
Revisado em: 22/08/2016
Aprovado em: 29/08/2016

1. INTRODUÇÃO

Visualização volumétrica consiste em um conjunto de técnicas de computação gráfica que permite analisar e explorar o interior de volumes (KAUFMAN, 2003). Essas técnicas fornecem ferramentas para diferentes áreas da ciência, tais como: Geologia, Engenharia Mecânica e, em especial, Medicina.

Ao longo dos anos, a adoção de ferramentas para visualização de volumes foi limitada pelo alto custo computacional exigido pelo processo de *rendering*. Desde as últimas duas décadas, novas tecnologias de hardware tornaram-se recursos cada vez mais utilizados para lidar com esse desafio (SANTOS et al., 2016). Contudo, antes mesmo dos avanços tecnológicos, pesquisadores já buscavam alternativas para redução do custo computacional. Um conjunto dessas alternativas são as técnicas de *Hybrid Volume Rendering* (HVR) (SANTOS et al., 2016).

Em um trabalho anterior (Eler et al, 2006), introduzimos a técnica *Volume on Surface* (VoS). A técnica é uma abordagem de HVR que combina o algoritmo *Ray Casting* com isosuperfícies para acelerar o processo de visualização de volumes. Mais recentemente (SANTOS et al., 2016), utilizamos CUDA para paralelizar a técnica VoS e assim, propiciar uma aceleração ainda maior do processo de *rendering*.

Neste trabalho, apresentamos a ferramenta HVRview (*Hybrid Volume Rendering Visualizer*). Essa ferramenta implementa duas versões da técnica VoS: uma paralela para execução em GPU e uma sequencial para execução em CPU. Em alguns testes, a abordagem paralela obteve *speedups* superiores a 24 vezes em relação à versão sequencial. Salientamos que avaliar a qualidade das imagens geradas e o desempenho da técnica VoS em relação a outras técnicas de visualização volumétrica não é escopo deste trabalho. Diferentes análises da abordagem já foram realizadas em nossos trabalhos anteriores (Eler et al, 2006; SANTOS et al., 2016).

Além desta introdução, este documento está organizado em outras seis seções. A Seção 2 apresenta os trabalhos relacionados. Conceitos bases de visualização volumétrica são apresentados na Seção 3. A Seção 4 descreve a técnica VoS. A ferramenta HVRview é apresentada na Seção 5. Os resultados são discutidos na Seção 6. Finalmente, a Seção 7 conclui este trabalho.

2. TRABALHOS RELACIONADOS

Diferentes ferramentas para visualização de volumes foram propostas na literatura. O HVRview se diferencia das demais ferramentas, ao utilizar uma técnica de HVR.

Voreen[1] *(Volume Rendering Engine)* é um projeto de código aberto iniciado em 2005 que permite a integração de diferentes técnicas de visualização de volumes. A ferramenta é implementada em C++, OpenGL e GLSL para execução paralela em GPU. Diferente do HVRview, o Voreen permite a utilização de funções de transferências multidimensionais e implementa técnicas de *Direct Volume Rendering*. ImageViS3D[2] é um software que foca em escalabilidade para permitir a exploração de grandes volumes, com até mesmo terabytes de dados. O software implementa o *Ray Casting* e, assim como o Voreen, permite a utilização de funções de transferências multidimensionais. No ImageViS3D, *widgets* são utilizados para auxiliar o usuário durante o ajuste dessas funções. VolView[3] é uma ferramenta de código aberto que também implementa o *Ray Casting* em GPU, no entanto foca em utilizar estratégias para a exploração de *datasets* médicos.

3. VISUALIZAÇÃO VOLUMÉTRICA

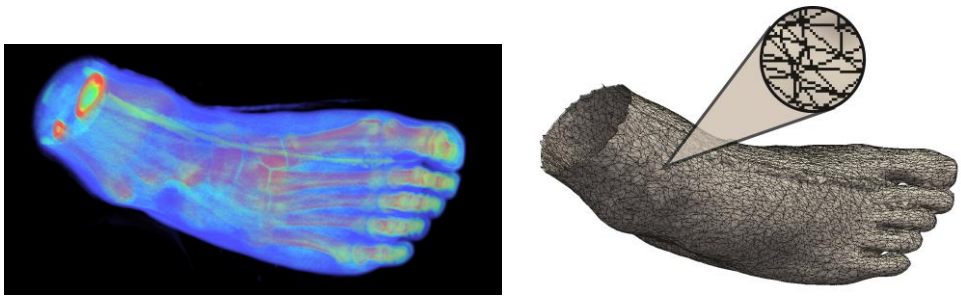
Técnicas de visualização volumétrica permitem investigar, de forma interativa, estruturas e regiões internas de um volume (KAUFMAN, 2003). De acordo com Elvins (1992), essas técnicas podem ser organizadas em duas categorias principais: *Surface-Fitting* (SF) e *Direct Volume Rendering* (DVR).

Técnicas da primeira categoria (e.g., *Marching Cubes* (LORENSEN, 1987)) operam sobre um modelo geométrico intermediário, formado por primitivas geométricas inseridas em um volume para a extração de superfícies (SEIXAS; GATTASS, 1999). Em comparação às técnicas de DVR, técnicas de SF são mais rápidas, mas não são adequadas a todas situações de exploração volumétrica. Em alguns casos, técnicas de SF não são capazes de permitir a visualização de todos os detalhes de um volume (ELER et al., 2006).

Técnicas de DVR operam diretamente sobre os dados volumétricos, incorporando características e detalhes originais de um *dataset*. Uma técnica de DVR varre todo o volume visualizado e é desta forma, capaz de gerar imagens de maior qualidade, mas mediante um maior custo computacional do que as técnicas de SF (ELER et al., 2006).

A Figura 1 apresenta o volume de um pé humano e uma isosuperfície extraída a partir desse volume. A imagem do volume (veja a Subfigura 1a) foi gerada pelo software *Kitware VolView,* que também foi utilizado para extração da superfície (veja a Subfigura 1b). Como pode ser observado, a estrutura da superfície extraída é formada por uma malha de triângulos.

[1] *Volume Rendering Engine* (Voreen). Disponível em: <http://www.voreen.org/>. Acesso em: 12 de agosto de 2016.
[2] ImageVis3D. Disponível em: <http://www.sci.utah.edu/software/imagevis3d.html>. Acesso em: 12 de agosto de 2016.
[3] Volview. Disponível em: <http://www.kitware.com/opensource/volview.html>. Acesso em: 12 de agosto de 2016.



a) Volume **b) Superfície**

Figura 1. Volume de um pé humano e sua superfície extraída.

O termo *Hybrid Volume Rendering* descreve métodos que integram diferentes estratégias de visualização volumétrica em uma única abordagem (ELER et al., 2006). Tipicamente, abordagens híbridas combinam técnicas de SF e DVR, visando agrupar vantagens de cada categoria (SANTOS et al., 2016).

4. VOLUME ON SURFACE

VoS (ELER et al., 2006) é uma técnica de HVR que combina o algoritmo *Ray Casting* com isosuperfícies para acelerar o processo de *rendering*. Essa técnica propicia a visualização de detalhes internos do volume através de sua isosuperfície extraída. O funcionamento básico do algoritmo consiste na utilização da malha da isosuperfície e de uma estrutura de cones virtuais para guiar o disparo de raios em direção ao volume. O algoritmo é dividido em quatro etapas: construção de cones, pré-visualização, atribuição de cor aos vértices e iluminação / projeção.

Na primeira etapa, é construído um conjunto de cones, com diferentes ângulos de abertura, em cada vértice da malha da superfície. O vetor normal do vértice é utilizado como eixo central dos cones. Um cone ainda é constituído por um conjunto de raios uniformemente distribuídos pela base circular, como apresentado na Figura 2.

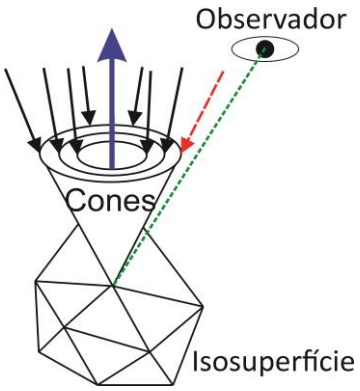


Figura 2. Exemplo de estruturas de cones na técnica *Volume on Surface*. Três cones virtuais atribuídos a um vértice.
Fonte: Santos et al. (2016).

Na etapa de pré-visualização, os raios são disparados dos vértices em direção ao volume. Após o disparo, os raios atravessam o volume. Ao longo do caminhamento de cada raio, ocorre um processo de acumulação de cor e opacidade. Esse processo compreende a utilização de funções de transferência. Uma função de transferência mapeia informações presentes nos dados volumétricos para propriedades

visuais e assim, determina as contribuições de cada amostra do volume atingida por um raio. No fim do processo de acumulação, há uma cor final armazenada em todos os raios e, consequentemente, um conjunto de cores associado a cada vértice da isosuperfície.

A terceira etapa do algoritmo é responsável por determinar qual raio atribuirá cor ao seu respectivo vértice. O algoritmo seleciona o raio (veja a seta vermelha na Figura 2) que mais se aproxima da linha de observação (veja a linha verde na Figura 2).

Na última etapa, algoritmos convencionais de *rendering* utilizam modelos de iluminação e projeção para gerar a imagem final. Durante essa etapa, um processo de interpolação determina a cor dos polígonos por meio das cores atribuídas aos seus vértices.

5. HYBRID VOLUME RENDERING VISUALIZER

Hybrid Volume Rendering Visualizer (HVRview) é uma ferramenta que utiliza a técnica VoS para visualização de volumes. Na ferramenta, uma versão paralela da técnica é implementada em CUDA, C++ e OpenGL para execução em GPU. Uma versão sequencial é implementada em CPU.

5.1 ORGANIZAÇÃO DOS DADOS

A ferramenta HVRview utiliza três conjuntos de dados principais: volume, isosuperfície extraída e cones virtuais. É importante destacar que o processo de extração de isosuperfícies a partir de um volume não faz parte do escopo da ferramenta. À vista disso, deve ser executado por um software externo.

No HVRview, um volume é originalmente formado por um conjunto de imagens (fatias) dispostas em sequência, com um espaçamento predefinido entre cada uma. A ferramenta transforma esse conjunto em uma malha volumétrica regular. Em visualização volumétrica, os elementos que compõem esse tipo de malha são denominados *voxels*. Na implementação, cada *voxel* armazena uma única informação: um valor escalar de intensidade que representa um tom de cinza. Durante a execução, a posição espacial de um *voxel* é calculada a partir da posição desse elemento na malha volumétrica.

A isosuperfície é constituída por uma malha de vértices, polígonos (triângulos) e vetores normais. Essa estrutura é originalmente descrita em um arquivo VTK[4] (*Visualization Toolkit*). Durante a interação do usuário, informações de cor são adicionadas aos vértices e aos polígonos.

Na estrutura de cones, cada cone consiste em um conjunto de raios. Na implementação, um raio é representado

[4] *Visualization Toolkit File Format*. Disponível em: <http://www.vtk.org/wp-content/uploads/2015/04/file-formats.pdf>. Acesso em: 04 de agosto de 2016.

como um segmento de reta e uma cor RGB. Além disso, uma das extremidades desse segmento é o próprio vértice ao qual o raio está associado.

5.2 ORGANIZAÇÃO DA FERRAMENTA

A ferramenta HVRview é composta por dois módulos: **módulo de interface de usuário** (MIU) e **módulo de *rendering*** (MR). Os módulos são executados em processos diferentes que se comunicam. Uma visão geral da organização da ferramenta HVRview é apresentada na Figura 3.

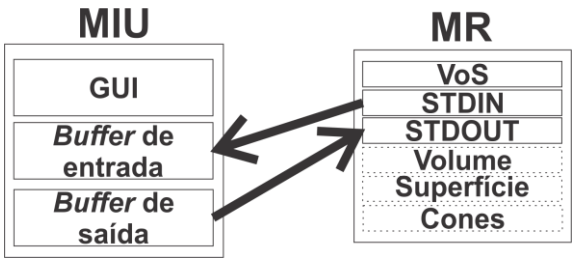


Figura 3: Organização da ferramenta.

O primeiro módulo é responsável pelas seguintes funções: capturar a interação do usuário e estabelecer comunicação com o MR. O MIU é implementado em Java e organizado em três *threads* que constituem um único processo. Uma primeira *thread* gerencia a interface gráfica do usuário (GUI). Sempre que o usuário interage com a GUI, a thread captura os dados necessários para que sejam transmitidos para o MR. No MIU, o processo de comunicação é estabelecido por dois componentes, os *buffers* de entrada e saída de dados. Uma *thread* coordena cada um dos componentes.

MR é o módulo principal do HVRview. O módulo realiza a execução do algoritmo VoS e utiliza um modelo de computação heterogênea. O MR é implementado em C/C++, CUDA C e OpenGL. A comunicação com o MIU é executada em CPU. Uma *thread* gerencia o arquivo STDIN (*Standard Input*), que se comunica com o *buffer* de saída do MIU. Uma segunda *thread* gerencia o arquivo STDOUT (Standard Output) e se comunica com o *buffer* de entrada do MIU. O MR ainda captura a interação do usuário quando a posição do observador em cena é modificada.

O algoritmo VoS paralelo é parcialmente implementado em CUDA. Uma descrição em maiores detalhes dessa implementação pode ser obtida em nosso trabalho anterior (Santos et al., 2016). As estruturas de volume, superfície e cones são armazenadas na memória da GPU. Cada uma das três primeiras etapas do algoritmo é implementada em *kernels* CUDA. No *kernel* da primeira etapa, uma *thread* constrói um raio virtual, estabelecendo uma posição espacial de uma extremidade desse raio. No segundo *kernel*, cada *thread* realiza o caminhamento de um raio em direção ao volume. No fim do processo, a cor resultante

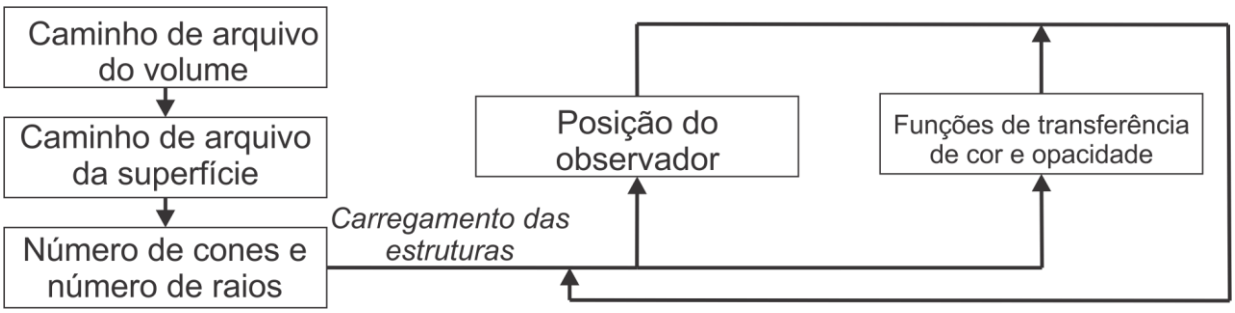


Figura 4. *Pipeline* de interação do usuário.

acumulada no raio é armazenada. No último *kernel*, cada *thread* realiza um produto escalar entre os raios de um vértice e a direção de observação para determinar o raio que atribuirá cor ao vértice. O raio mais próximo da linha de observação, i.e., o raio com menor produto escalar é selecionado. A última etapa é implementada em OpenGL. O algoritmo utiliza os processos de iluminação e projeção para determinar as cores dos polígonos e assim, gerar a imagem final. A implementação sequencial do algoritmo VoS é semelhante a implementação paralela. Entretanto, cada *kernel* CUDA é substituído por um laço de repetição, no qual cada iteração desempenha as mesmas funções que uma *thread* desse *kernel*.

5.3 *PIPELINE* DE INTERAÇÃO DO USUÁRIO

O *pipeline* de interação do usuário pode ser observado na Figura 4. Em um primeiro momento, é necessário informar os caminhos de arquivo dos dados de volume e superfície. Em seguida, o usuário deve introduzir os ângulos de abertura dos cones e o número de raios por cone. Todos os dados são transmitidos ao MR que carrega e inicia as estruturas do volume, superfície e cones. Um histograma do volume é transmitido do MR ao MIU.

No HVRview, as funções de transferência de cor e opacidade são unidimensionais. Um exemplo da interface de especificação dessas funções é apresentado na Figura 5.

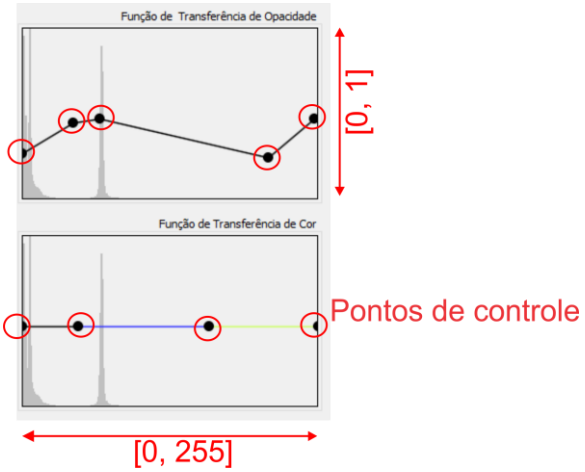


Figura 5: Interface de especificação de funções de transferência de cor e opacidade no HVRview.

A interação do usuário ocorre de forma manual. Em ambas as funções, é necessário definir um conjunto de pontos de controle que delimitam intervalos de cor ou opacidade. Na interface, o eixo das abcissas compreende intensidades no intervalo de

Tabela 1. Especificação de volume e superfície.

	Volume	Superfície	
		Número de vértices	Número de polígonos
Bucky Ball	32x32x32	6421	11836
Dente humano	256x256x161	63698	127400

valores inteiros [0, 255]. Na função de opacidade, o eixo das coordenadas compreende um intervalo de valores reais [0, 1]. Na função de cor, uma cor específica pode ser atribuída a cada intervalo. Para guiar o usuário, o plano de fundo de cada interface de especificação apresenta um histograma do volume utilizado.

6. RESULTADOS

Nos testes, foi utilizado um computador com a seguinte especificação: CPU Intel Core 2 Quad Q6700, 4 GB RAM, GPU NVIDIA GT 740 com 364 CUDA *cores* e Sistema Operacional Windows 8.1 64 bits. Em relação à configuração dos cones, foram utilizados 4 raios por cones e 4 cones com os seguintes ângulos de abertura: 15°, 60°, 105° e 160°. Todos os resultados apresentados representam uma média de 5 execuções. As especificações do volume e da superfície dos *datasets* utilizados nos testes são apresentadas na Tabela 1.

A Figura 6 apresenta imagens dos *datasets Bucky Ball*[5] e dente humano[6] geradas pela ferramenta HVRview.

Para avaliar o desempenho da ferramenta em relação às execuções paralela em GPU e sequencial em CPU, foram definidas duas situações de interação do usuário. Na primeira situação, a posição do observador é rotacionada ao redor do eixo x. Na segunda, além da rotação do observador, as funções de transferência são ajustadas. A taxa de quadros do HVRview nessas duas situações de interação é apresentada na Tabela 2. Em média, a paralelização em GPU propiciou um *speedup* de 12,1 vezes na primeira situação. Além disso, quando as funções de transferência também são ajustadas, o aumento de desempenho foi ainda maior, proporcionando um *speedup* médio de 50,7 vezes.

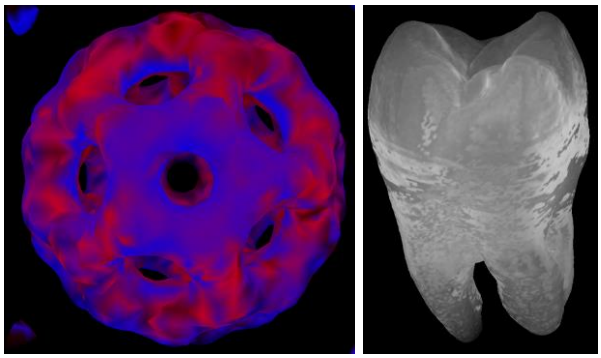


Figura 6. Imagens dos volumes *Bucky Ball* e dente humano geradas pelo HVRview**.**

[5] *Dataset* de volume *Bucky Ball*. Disponível em: http://www9.informatik.uni-erlangen.de/External/vollib/>. Acesso em: 15 de abril de 2015.

[6] *Dataset* de volume dente humano. Disponível em: http://www9.informatik.uni-erlangen.de/External/vollib/>. Acesso em: 15 de abril de 2015.

Tabela 2. Taxa de quadros por segundo (fps) e *speedup* no HVRview**.**

	Modificação da posição do observador			**Ajuste de funções de transferência**		
	CPU	GPU	*Speedup*	CPU	GPU	*Speedup*
Bucky Ball	44,5 fps	226,8 fps	5,1	2,9 fps	71,7 fps	24,7
Dente humano	4,5 fps	86,1 fps	19,1	0,03 fps	2,3 fps	76,7

CONCLUSÃO

A ferramenta HVRview é uma alternativa aos softwares de visualização de volumes presentes na literatura, visto que implementa uma abordagem de HVR, a técnica VoS.

Neste trabalho, avaliamos as versões sequencial em CPU e paralela em GPU da técnica VoS. Com base nos testes realizados, foi possível constatar que a utilização de GPUs possibilitou um efetivo aumento de desempenho.

A implementação em CUDA restringe a execução paralela do HVRview apenas às GPUs produzidas pela NVIDIA. Assim, é interessante que um trabalho futuro investigue o desempenho da ferramenta em outras plataformas de hardware, compreendendo não só a execução paralela em GPU, mas também em CPU.

Funções de transferência são componentes fundamentais do processo de exploração volumétrica. Nesse escopo, funções de transferência multidimensionais e estratégias mais sofisticadas para interface de especificação dessas funções são importantes melhorias que devem ser implementadas na ferramenta.

REFERÊNCIAS

ELER, D. M. et al. **Empowering iso-surfaces with volume data**. *Proceedings of GRAPP International Conference on Computer Graphics Theory and Applications*, 2006.

ELVINS, T. T. **A survey of algorithms for volume visualization**. *SIGGRAPH Comput. Graph.*, ACM, New York, NY, USA, v. 26, n. 3, p. 194–201, ago. 1992. ISSN 0097-8930. http://doi.acm.org/10.1145/142413.142427

KAUFMAN, A. E. **Volume Visualization: Principles and Advances**. 2003.

LORENSEN, W. E.; CLINE, H. E. **Marching cubes: A high resolution 3D surface construction algorithm**. In: *ACM siggraph computer graphics*. ACM, 1987. p. 163-169. https://doi.org/10.1145/37401.37422

SANTOS, R. S. et al. **An evaluation of the Volume on Surface approach**. To appear in: *2016 29th SIBGRAPI Conference on Graphics, Patterns and Images*. [S.l.: s.n.], 2016. https://doi.org/10.1109/sibgrapi.2016.066

SEIXAS, R. B.; GATTASS, M. **Introdução à Visualização Volumétrica**. Monografia em Ciência da Computação, v. 19991, 1999.