EXTRAÇÃO DE SILHUETAS EM DADOS VOLUMÉTRICOS

Aluno: Rodrigo Arruda Torres
Orientador: Sinésio Pesco

Introdução

O trabalho com dados volumétricos é cada vez mais comum em diversas áreas do conhecimento, especialmente em aplicações ligadas à ciência e à medicina. Estes dados podem ser obtidos através de escaneamento de fenômenos reais ou simulações computacionais.

Com o desenvolvimento de novas tecnologias de escaneamento e com o aumento da complexidade dos algoritmos de simulação, o tamanho dos dados gerados tem crescido, impondo a necessidade de desenvolver métodos e algoritmos capazes de reduzir o tempo de visualização para dados volumétricos de grande porte.

Uma das possibilidades de visualização de dados volumétricos consiste na extração das chamadas curvas de silhueta. Neste trabalho foi desenvolvido um algoritmo baseado no trabalho de Burns et al. [1], capaz de realizar tal extração em dados de grande porte, obtendo resultados bastante satisfatórios em termos de velocidade de processamento e interação. Para melhorar a qualidade de visualização da silhueta, utilizou-se um método de remoção de linhas não visíveis, conforme descrito em [2].

Objetivos

Desenvolver um algoritmo computacional em linguagem C, para a extração de curvas silhuetas em dados volumétricos de grande porte (256^3 e 512^3). Aplicar o método de oclusão implícita para a remoção de curvas silhuetas não visíveis. Além disso, estudar e otimizar os tempos de processamento e interação de ambos os algoritmos.

Trabalhos Anteriores

Dentre os diversos trabalhos, na área de Computação Gráfica, que tratam da manipulação de dados volumétricos, um dos mais utilizados é o algoritmo Marching Cubes [3] para geração de isosuperfícies através de uma malha de triângulos. Para a obtenção da curva silhueta, aplicamos o algoritmo Marching Lines [5], capaz de extrair linhas características dentro da triangulação gerada pelo Marching Cubes. Ambos os algoritmos são importantes alicerces para os modelos desenvolvidos neste trabalho e, portanto, seus funcionamentos básicos serão explicados em seção posterior.

Burns et al. [1] descrevem um algoritmo para geração de curvas silhuetas em um dado volumétrico regular, utilizando o algoritmo Marching Lines. Neste trabalho eles apresentam diferentes estratégias para se obter um número significante de curvas silhuetas sem percorrer todo o grid (ver seção *Conceitos Básicos* para definição de grid). Uma das dificuldades no trabalho de Burns é visualizar as curvas silhuetas com remoção de linhas interativamente.

Wilhelms e Van Gelder [6] utilizam uma octree para otimizar a extração da isosuperfície, evitando percorre todo o grid. Pesco et al. [4] e Lisowski [2] apresentam o método de oclusão implícita e aplicações em visualização volumétrica: A partir do trabalho de Wilhems e Gelder [6], utilizam-se os nós positivos e os nós negativos para determinar uma região de oclusão realizando um estudo da primeira mudança de sinal dos nós da O*ctree*.

Conceitos Básicos

Um primeiro aspecto a ser destacado é o fato de que dados volumétricos regulares, como aqueles estudados neste trabalho, podem ser divididos em estruturas mais simples (cubos ou *voxels*), de modo que juntos formem uma estrutura mais complexa conhecida como grid (ver figura 1). A cada vértice de cada cubo está associado um valor real, definindo, assim, um campo escalar discreto $f : D \subset R^3 \rightarrow R$. Da mesma forma, podemos afirmar que, a cada isovalor *w* existe uma isosuperfície associada, definida por $f^{-1}(w)$.

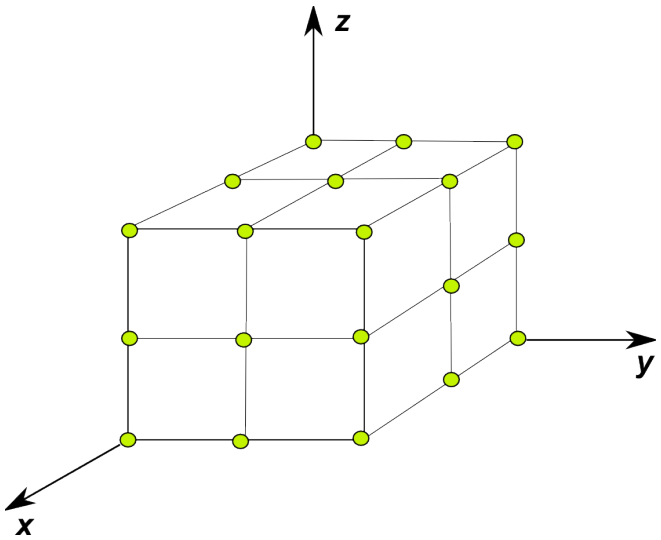

Figura 1 Dado Volumétrico Regular.

Um segundo importante conceito que deve ser abordado é a definição de silhueta. Uma curva silhueta sobre uma isosuperfície S é tal que para cada ponto p pertencente à curva vale $N_p.V = 0$ (produto escalar), sendo N_p o vetor normal à isosuperfície e V o vetor que representa o observador. Em outras palavras, a silhueta de um dado volumétrico associada a certa isosuperfície representa o contorno da mesma. Abaixo, a figura 2 ilustra uma curva silhueta.

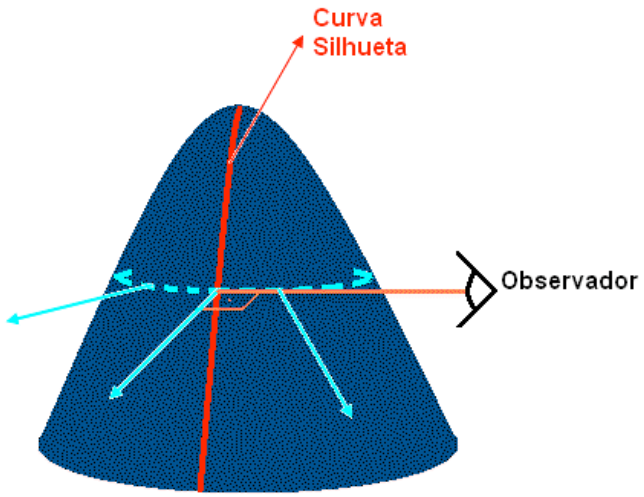

Figura 2 Esquema ilustrativo da definição de curva de silhueta

Marching Cubes e Marching Lines

O algoritmo Marching Cubes é usado para renderizar dados volumétricos, construindo isosuperfícies a partir de campos escalares em três dimensões. Seu funcionamento básico consiste em classificar os oito vértices de cada cubo ou *voxel* como sendo positivo ou negativo em relação ao isovalor de interesse. Se o campo escalar associado ao vértice for maior do que o isovalor, então o vértice é classificado como positivo. Se for menor, então é classificado como negativo.

Desta forma, gera-se uma espécie de código que representa a situação do cubo analisado. Este código é comparado com uma das 256 possíveis combinações (que podem ser reduzidas a 16, considerando as simetrias), de modo que seja proposta a triangulação que melhor represente a isosuperfície interna ao cubo. Os vértices da triangulação são determinados através de interpolação linear, bem como os campos escalares associados a eles, conforme ilustrado na figura 3 (a).

Já o algoritmo Marching Lines é usado para obtenção de curvas características dentro da triangulação gerada pelo algoritmo Marching Cubes. No caso da extração de silhuetas, tema deste trabalho, o funcionamento do Marching Lines tem início com o cálculo dos produtos escalares, em cada vértice da triangulação, entre os vetores citados na seção *Conceitos Básicos*. Em seguida, verifica-se a ocorrência de mudança de sinal entre os valores encontrados para cada par de vértices de cada triângulo. Em caso afirmativo, é garantido que, ao longo da aresta que os conecta, existe um ponto cujo produto escalar a ele associado vale zero, ponto este que faz parte da curva silhueta e é obtido via interpolação linear (ver figura 3 (b)). Este procedimento é realizado para toda a triangulação e os pontos assim determinados, ao serem conectados, formam a curva silhueta interna ao cubo analisado.

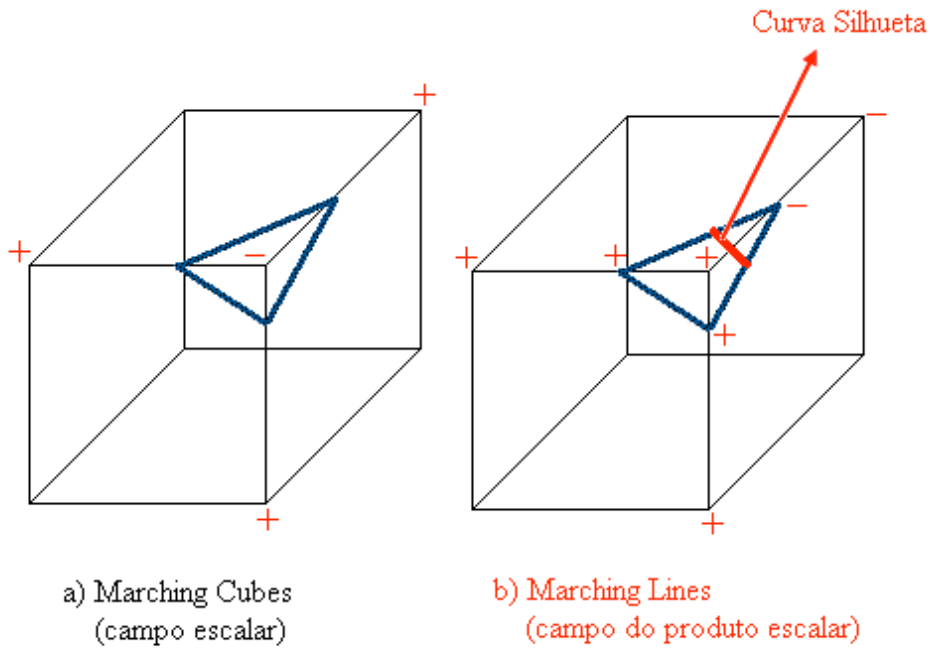

Figura 3 Marching Cubes e Marching Lines

Algoritmo de Extração de Silhuetas – Pré-classificação de cubos

O algoritmo de extração proposto neste trabalho pode ser dividido em duas etapas. A primeira consiste na obtenção da silhueta em relação à posição inicial do observador, enquanto a segunda consiste na obtenção da mesma para o caso em que ocorre mudança na posição do observador.

A fim de reduzir o tempo de processamento associado à primeira etapa do código, é realizada uma pré-classificação dos cubos que compõem o grid [6]. Isto possibilita que menos cubos sejam analisados, contribuindo para a redução no tempo proposta.

Esta pré-classificação faz uso da estrutura de dados conhecida como *Octree*. Uma *Octree* é uma estrutura em árvore na qual cada nó que não seja folha está ligado a outros oito nós. No caso em que é usada para análise de dados volumétricos, a sua raiz representa todo o grid, enquanto os oito filhos desta raiz representam os oito cubos do primeiro nível de subdivisão do grid e assim por diante. Por exemplo, um grid de tamanho 256^3 corresponde a uma *Octree* de nível oito, pois $2^8 = 256$.

A *Octree* usada para pré-classificar os cubos armazena em cada um de seus nós os valores máximo e mínimo dos campos escalares do "sub-grid" que representa. A determinação destes valores é feita no sentido inverso à formação da estrutura, ou seja, tem início nos nós que representam apenas um cubo e terminam no nó raiz que representa o grid inteiro.

Uma vez conhecidos estes valores máximo e mínimo de cada nó, é possível classificá-los em relação ao isovalor de interesse. Se o isovalor for maior do que o valor máximo, então aquele nó é classificado como negativo. Se for menor do que o valor mínimo, então é classificado como positivo. Caso o isovalor esteja entre o máximo e o mínimo, então o nó é classificado como "zero". Nos dois primeiros casos, é garantido que a silhueta não está presente, uma vez que não há cubos contendo o isovalor procurado.

Com este procedimento, descarta-se uma grande quantidade de nós que certamente não apresentam a silhueta. Apenas os cubos pré-classificados como "zero" são passíveis de conterem as curvas. Portanto, atinge-se o objetivo de, através de um pré-processamento, reduzir o tempo de extração da silhueta para a posição inicial do observador.

O Algoritmo de Extração de Silhuetas

Como mencionado, o algoritmo desenvolvido divide-se em duas etapas. Na primeira, ele recebe como parâmetros de entrada os índices (i, j, k) relativos às coordenadas de um cubo inicial, classificado como "zero" conforme a pré-classificação descrita. Conhecidas tais coordenadas, é possível reconstruir este cubo, ou seja, determinar os campos escalares associados a cada um de seus vértices, através de uma função auxiliar a qual acessa o dado volumétrico estudado. Em seguida, é feito um primeiro teste para verificar a presença de silhueta interna a este cubo. Caso todos os campos escalares sejam maiores ou menores do que o isovalor de interesse, então o cubo é descartado e o algoritmo é reiniciado com um outro cubo pré-classificado como "zero". Caso contrário, o algoritmo prossegue.

A próxima etapa consiste na determinação das faces que compõem a isosuperfície interna ao cubo, utilizando-se, para tal, o algoritmo Marching Cubes, conforme descrito anteriormente. Em seguida, com base nos conceitos propostos pelo algoritmo Marching Lines, também já descrito, extrai-se a silhueta presente no cubo.

Ao final desta etapa, é iniciada uma outra, na qual dois procedimentos são realizados. O primeiro consiste no armazenamento das coordenadas deste cubo. Já o segundo consiste na determinação da face de saída da curva silhueta.

São dois os motivos para o armazenamento das coordenadas. Por um lado, elas são necessárias para determinar o momento do fechamento da curva de silhueta. Caso, ao longo do algoritmo, este cubo inicial seja reencontrado, então a curva foi fechada e deve-se buscar a

extração de outras curvas a partir de outras sementes. Por outro lado, a segunda etapa do código beneficia-se de tal armazenamento. Os cubos nos quais é garantida a presença de silhueta servirão de sementes para a extração das novas curvas que devem ser geradas quando há mudança na posição do observador. Da mesma forma que a pré-classificação, este procedimento permite um aumento na velocidade de processamento, porque restringe a passagem do código a um número menor de sementes.

Quanto à determinação da face de saída da silhueta, novamente objetiva-se ganhos na velocidade de processamento. Ao longo do processo de triangulação, é guardada a face pela qual a curva pretende sair do cubo. Claramente, existem momentos em que a curva é interna ao cubo e, portanto, não há face de saída. Porém, ao término da triangulação, pode-se garantir que existe uma face de saída, pois a curva não pode ficar incompleta dentro do cubo. Realiza-se, então, um teste que verifica se há incoerência na face de saída obtida, isto é, se ela é igual à face de entrada. Neste caso, sabe-se que a primeira face armazenada ao longo da triangulação é, na verdade, a de saída.

Saber a face de saída gera ganhos de processamento porque permite determinar qual é o próximo cubo que deve ser analisado. Com isso, não é necessário percorrer toda a vizinhança do mesmo, partindo-se diretamente para o vizinho em que é garantida a existência de silhueta.

O algoritmo é, então, repetido para o novo cubo assim encontrado, até que a curva se feche – como mencionado – ou um bordo seja encontrado. Neste último caso, retorna-se à primeira semente e percorre-se a curva no sentido contrário, evitando perda de qualquer porção da silhueta.

O tratamento do caso em que há mudança na posição do observador, como dito, é feito de forma diferente. Leva-se em consideração, para tal, a idéia de que pequenas variações na posição do observador não acarretam grandes variações nas curvas silhueta, isto é, não há o surgimento de novas curvas significativas em cubos que não tenham sido visitados na primeira passagem. Para a extração da silhueta, portanto, utiliza-se a coleção de sementes construída ao longo da primeira etapa.

Para remoção das linhas silhuetas não visíveis, foi incorporado um método de oclusão baseado no conceito de oclusão implícita [2]. A oclusão implícita determina regiões que não são visíveis ao observador por estarem encobertas pela isosuperfície, e as silhuetas contidas nessas regiões são marcadas como não visíveis pelo buffer de profundidade. Esse processo é baseado unicamente na mudança de sinal dos nós da octree previamente classificados. Sendo assim, a isosuperfície não é explicitamente gerada, o que demanda pouco tempo adicional como foi observado nos resultados de seção 7.

Resultados

O algoritmo de extração das silhuetas pode se avaliado em relação a dois critérios. O primeiro é a eficiência na extração propriamente dita, ou seja, é a capacidade do algoritmo obter todas as curvas silhuetas significativas, sem levar em consideração, neste momento, se são ou não visíveis para o observador. O segundo é o tempo de processamento necessário para tal. Nestes dois quesitos, o algoritmo mostrou-se bastante eficaz, o que pode ser inferido através dos três exemplos abaixo dispostos (figuras 4, 5 e 6) e através dos tempos de processamento destacados nas tabelas 1 e 2.

Vale destacar, em relação aos resultados da extração, que a segunda etapa do algoritmo, responsável por fazê-las após a mudança na posição do observador, apresentou resultados ainda melhores do que os da primeira etapa, que faz uso dos cubos pré-classificados pela estrutura da *Octree*. Isto mostra que a estratégia de armazenamento dos cubos visitados na primeira etapa, para que sejam usados como sementes na segunda, é bastante importante e permite uma melhor visualização das curvas obtidas de maneira interativa.

Já o algoritmo de oclusão também pode ser avaliado segundo dois critérios. O primeiro diz respeito à sua capacidade de ocultar silhuetas que não sejam visíveis para o observador, em um processo de "limpeza" da imagem que permite sua melhor visualização e entendimento. O segundo diz respeito ao tempo de processamento acrescido quando a oclusão está ativada. Novamente, os resultados foram extremamente satisfatórios. Como pode ser visto nos exemplos, a oclusão mostrou-se eficiente em retirar as curvas ocultas sem que haja praticamente nenhum aumento significativo no tempo de processamento (tabela 3). Novamente, isto possibilita a manipulação interativa do resultado obtido.

Figura 4 Extração da silhueta e oclusão das curvas ocultas para dado volumétrico BONSAI (256^3) – à esquerda, o resultado da extração e à direita o resultado após ativação da oclusão.

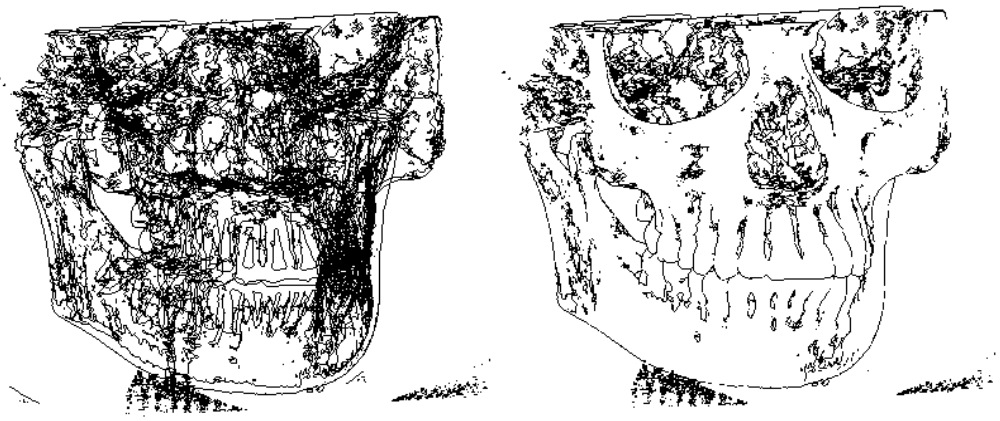

Figura 5 Extração da silhueta e oclusão das curvas ocultas para dado volumétrico SKULL (256^3) – à esquerda, o resultado da extração e à direita o resultado após ativação da oclusão.

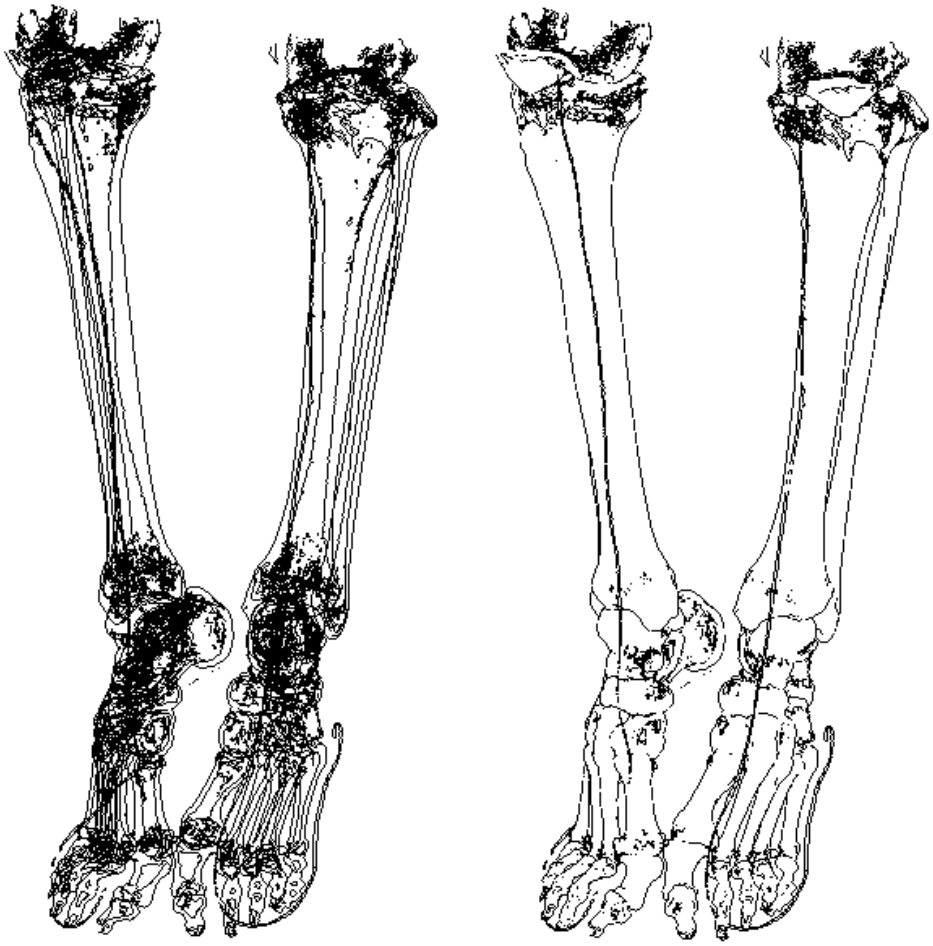

Figura 6 Extração da silhueta e oclusão das curvas ocultas para dado volumétrico FOOT (512^3) – à esquerda, o resultado da extração e à direita o resultado após ativação da oclusão.

Dado Volumétrico	Tempo médio de extração para posição inicial do observador (s)	Quadros por segundo (FPS – Frames per second)
BONSAI	0,251	3,98
SKULL	0,625	1,60
FOOT	0,516	1,94

Tabela 1 Tempos médios para extração da silhueta para a posição inicial do observador a partir dos cubos pré-classificados.

Dado Volumétrico	Tempo médio de extração após mudança na posição do observador sem ativar a oclusão (s)	Quadros por segundo (FPS – Frames per second)
BONSAI	0,0395	25,3
SKULL	0,141	7,09
FOOT	0,128	7,81

Tabela 2 Tempos médios para extração da silhueta após mudança na posição do observador.

Dado Volumétrico	Tempo médio para ativação e funcionamento da oclusão (s)	Tempo médio de extração após mudança na posição do observador sem ativar a oclusão (s)	Tempo total de visualização da silhueta com oclusão ativada (s)	Quadros por segundo (FPS – Frames per second)
BONSAI	0,00450	0,0395	0,0469	21,3
SKULL	0,0109	0,141	0,151	6,62
FOOT	0,00803	0,128	0,137	7,30

Tabela 3 Tempos médios para ativação e funcionamento da oclusão e tempos médios totais de visualização da silhueta com oclusão ativada.

Conclusões

Neste trabalho foram desenvolvidos dois algoritmos de análise de dados volumétricos: um responsável pela extração de curvas silhuetas e outro responsável por esconder as curvas não visíveis para o observador.

Ambos os casos foram testados em dados volumétricos de grande porte (256^3 e 512^3), obtendo resultados bastante eficazes em termos de eficiência na extração e na oclusão, respectivamente, e nos tempos de processamento. Desta forma, o trabalho atingiu objetivo proposto de desenvolver métodos que fossem ao encontro da expectativa de se tratar de forma interativa, ou parcialmente interativa, dados de grande porte.

Agradecimentos

Os autores deste trabalho gostariam de agradecer ao CNPq pelo apoio financeiro.

Referências

[1] BURNS, M.; KLAWE, J.; RUSINKIEWICZ, S.; FINKELSTEIN, A.; DeCARLO, D. Line Drawings from Volume Data. **ACM Transactions on Graphics (Proc. SIGGRAPH)**, v.24, n.3, p. 512-518, aug. 2005.

[2] LISOWSKI, K.S.L., Método da Oclusão Implícita e suas Aplicações em Visualização, Dissertação de Mestrado, Departamento de matemática – PUC-Rio, Março 2007.

[3] LORENSEN, W. E.; CLINE, H. E. Marching Cubes: a high resolution 3D surface construction algorithm. **Computer Graphics (Proc. SIGGRAPH),** v.21, n. 4, p. 163-169, 1987.

[4] PESCO, S., LINDSTROM, P., PASCUCCI, V., SILVA, C. Implicit Occluders. **IEEE/SIGGRAPH Symposium on Volume Visualization 2004**, Austin/Texas, October 2004, p. 47-54.

[5] THIRION, J.P.; GOURDON, A. The 3d marching lines algorithm. **Graphical Models and Image Processing***, v. 58*, n. 6, p. 503–509, nov.1996.

[6] WILHELMS J.; VAN GELDER A., Octrees for Faster Isosurface Generation, **ACM Trans. Graphics**, Vol. 11, No. 3, July 1992, pp. 201-227.