Operações Booleanas na Modelagem por Pontos

Heloisa Reis Leal[1], Waldemar Celes[2], Luiz Velho[1]

[1]IMPA – Instituto de Matemática Pura e Aplicada
Estrada Dona Castorina, 110 – 22460-320 Rio de Janeiro, RJ

[2]Departamento de Informática – Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente, 225, Gávea – 22453-900 Rio de Janeiro, RJ

heloreis@yahoo.com, celes@inf.puc-rio.br, lvelho@impa.br

Abstract. This article presents an overview of point-based representation and of two recent works that explore interactive boolean operations on point-based models: the work by Adams and Dutré and the work by Pauly et. Al.. Also presents the implementation of the algorithm proposed by Adams and Dutré with some improvements, and compares this implementation with the work by Pauly et. al..

Resumo. Este artigo apresenta uma breve visão de modelagem por pontos e de dois trabalhos que exploram as operações booleanas neste tipo de modelagem: o trabalho de Adams e Dutré e o trabalho de Pauly et. al. Apresenta também a implementação do algoritmo proposto em Adams e Dutré com algumas melhorias e uma comparação com o método de Pauly et. al..

1. Introdução

A modelagem por pontos começou a ser largamente explorada nesta década e trata-se de uma área muito promissora em computação gráfica dadas às várias e grandes vantagens em relação aos outros tipos de representação como malhas poligonais. Dentre as principais vantagens temos: grande simplicidade e flexibilidade, boa performance, robustez. Mas, o fator que mais impulsionou o seu grande desenvolvimento nesta década, é que ela é a melhor opção para tratar os dados de saída de equipamentos de aquisição de modelos 3D. O uso destes equipamentos tem aumentado muito, sua tecnologia tem progredido bastante e eles têm se tornado mais baratos. A tendência é que o uso destes equipamentos se torne o mais utilizado meio de obtenção de modelos 3D.

No trabalho com modelos 3D, freqüentemente são usadas as operações booleanas, que são operações de união, interseção ou diferença para criar novos modelos ou para alterá-los. A simplicidade da representação por pontos permite que as operações booleanas sejam realizadas com facilidade se compararmos com a realização destas em outros tipos de representação. Dois recentes trabalhos exploram estas operações na modelagem por pontos: o trabalho de Adams e Dutré[2] e o trabalho de Pauly et. al[3]. As contribuições dadas por este trabalho são: um breve visão sobre modelagem por pontos; apresentação dos trabalhos de Adams e Dutré e do trabalho de Pauly et. al.; propostas de melhorias ao algoritmo proposto por Adams e Dutré e a comparação dos métodos de Pauly e de Adams e Dutré.

2. Modelagem por Pontos

Um modelo por pontos representa um objeto através de seu contorno. É um tipo de representação B-Rep (Boundary Representation). Este contorno consiste em uma superfície 2D posicionada no espaço 3D. Na modelagem por pontos, ela é representada por um conjunto de pontos 3D que são uma amostragem dos (infinitos) pontos que formam a superfície contínua (Figura 1a).

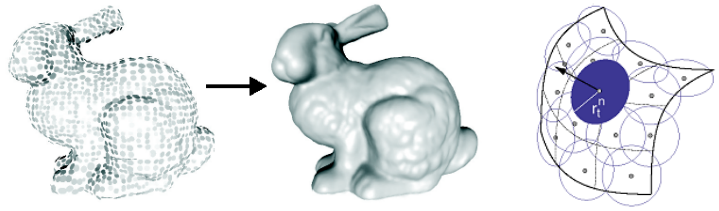

Figura 1: a) Modelo por pontos e sua visualização b) surfels na superfície

Estes pontos são denominados *surfels* e podem conter, além da posição 3D, outras informações como cor e coordenadas de textura. Se a amostragem não for uniforme, o *surfel* contém a informação do raio de alcance que é o raio de um disco cujo centro é a posição do *surfel*. O *surfel* pode também conter o vetor normal à superfície e assim o surfel pode ser visto como um disco orientado (Figura 1b).

Esta amostragem pode ser obtida por um processo de aquisição que consiste em obter uma amostragem de pontos da superfície do objeto real por meio de equipamentos como, por exemplo, o scanner 3D. Este processo pode ser dividido em três partes: aquisição da geometria, aquisição da textura e correção de falhas da aquisição (a aquisição da geometria deixa freqüentemente buracos no modelo devido a ruídos no processo ou à impossibilidade de aquisição de determinadas partes de uma superfície).

Para visualização de um modelo por pontos, a idéia inicial foi construir uma malha de polígonos a partir dos *surfels* e renderizá-la. Mas esta construção de malhas a partir dos *surfels* é um processo muito caro e lento, principalmente para modelos complexos. Na tentativa de obter uma visualização interativa estão sendo estudadas técnicas que utilizam *surfels* como primitiva de visualização. Segundo estas novas técnicas, a visualização do modelo por pontos pode ser dividida nas seguintes etapas: projeção em perspectiva de cada ponto do modelo na tela (análogo à projeção de vértices dos triângulos na malha triangular); processo de *shading* realizado para cada ponto visível; e por último, a imagem deve ser reconstruída sem buracos, pois podem haver pixels que não "receberam" projeção de nenhum ponto.

3. Operações Booleanas com Pontos

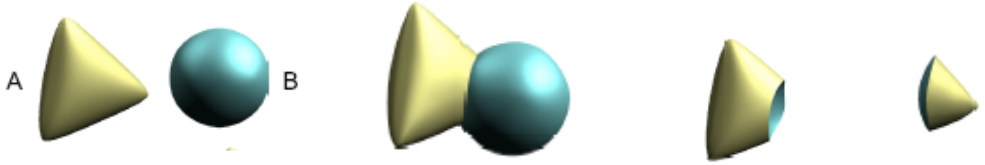

Figura 2: a. objetos A e B b. A U B: $A_f + B_f$ c. A – B: $A_f + B_d$ d. A ∩ B: $A_d + B_d$

Sejam A, B(Figura 2), objetos 3D sólidos modelados por pontos, temos os seguintes elementos: A_f (superfície de A que está fora de B); A_d (superfície de A dentro de B); B_f (superfície de B fora de A); B_d (superfície de B dentro de A). O resultado das operações booleanas entre A e B estão mostradas na Figura 2.

Lembrando que estas superfícies são conjuntos de *surfels,* para determinar os elementos acima é necessária a classificação de um *surfel* (que pode se resumir à classificação de um ponto que é o seu centro) de uma superfície como "dentro" ou "fora" de um outro modelo. Tanto o método de Pauly como o de Dutré utilizam a seguinte regra para esta classificação (que chamamos neste trabalho de Regra 2):

Regra 2: Dado um ponto x qualquer, um volume V determinado por uma superfície de contorno S, uma amostragem P da superfície S, um ponto $p \in P$ que é a amostra mais próxima a x, um vetor n_p normal à S em p e orientado para fora de S, temos que: $(x - p) \cdot n_p > 0 \Leftrightarrow x \notin V$.

Ou seja, para classificar um ponto x em relação a um modelo, verificamos o produto interno entre a normal em p (surfel mais próximo a x) e o vetor (x - p): se este produto interno for maior que zero, x não pertence ao volume V e será classificado como "fora"; senão será classificado como "dentro". Mais informalmente: se a normal em p apontar para a direção oposta a x, x será classificado como "dentro", senão "fora".

4. Método de Pauly

O modelo de Pauly é uma estrutura híbrida de pontos com a representação implícita *Moving Least Squares* - MLS. Esta estrutura é usada para fazer operações booleanas, e também operações de *free-form* (não abordadas neste trabalho). Para operações booleanas, a representação MLS permite que a curva de interseção entre os modelos seja reconstruída e *surfels* sejam colocados nesta. O algoritmo classifica *surfels* de um modelo como "dentro" ou "fora" de outro modelo. Pauly estende esta classificação para modelos que não são sólidos. Ele considera que mesmo um modelo não sólido divide o espaço da cena (espaço em que se encontra o modelo) em dois sub-espaços: dentro (os vetores normais apontam para direção oposta a este sub-espaço) ou fora do modelo (os vetores normais apontam para este sub-espaço). Para determinar esta classificação, Pauly utiliza a Regra 2, pesquisando o *surfel* mais próximo. Para pesquisa do *surfel* mais próximo, o método usa a estrutura *kd-tree* e também o algoritmo ANN e uma esfera que contém pontos com a mesma classificação.

5. Método de Adams e Dutré

Este método utiliza uma *octree* que subdivide o espaço do modelo em três: fora, dentro ou no contorno do modelo. Estes espaços são determinados pela classificação dos nós da *octree* a partir da Regra 2 acima citada.

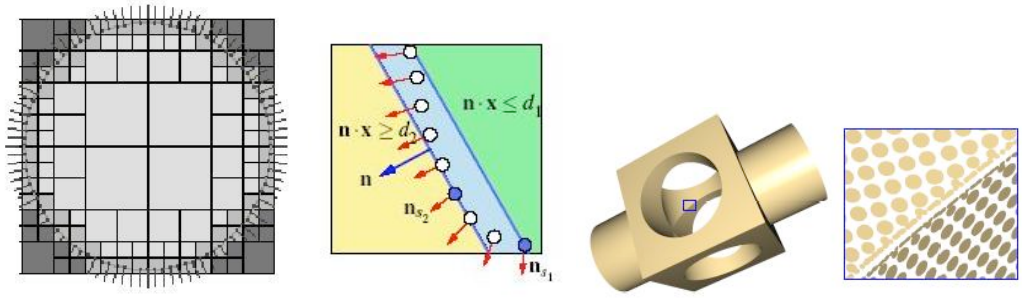

Figura 3: a) quadtree classificada b) nó subdividido c) tratamento da 'aresta viva'

É criada uma *octree* para cada modelo. Os nós da *octree* são, então, classificados como "dentro", "fora" e "contorno" (nó que contém *surfels*). O nó vazio é classificado por

um de seus vizinhos. Se só possui vizinhos vazios, ele fica com a classificação destes, senão o nó é classificado de acordo com o *surfel* do nó vizinho que está mais próximo ao nó a ser classificado. Se a normal deste *surfel* aponta para o nó vazio a ser classificado, então ele é classificado como “fora”, senão ele é classificado como “dentro”. Em geral, os nós “contorno” são subdivididos em três partes: uma que está fora do modelo, uma que está dentro do modelo e uma terceira que contém os *surfels* (Figura 4 b). Para classificar A em relação a B, inicialmente é feita uma avaliação da posição de A em relação a B. A e B podem estar nas seguintes situações: A não intersecta B; A intersecta apenas nós “fora” de B; A intersecta apenas nós “dentro” de B; A intersecta nós “fora” e nós “dentro” de B. No primeiro e no segundo caso, todos os *surfels* de A são classificados como “fora”; no terceiro caso, como “dentro” e no quarto caso, temos que descer a árvore classificando seus nós filhos.

Os *surfels* de A que intersectam *surfels* de B são classificados como “interseção”. Se eles não forem incluídos no modelo resultado, o sólido resultado terá buracos. Se eles forem incluídos sem algum tratamento, eles não representarão adequadamente a região. O tratamento de interseção proposto por Adams e Dutré não só evita os buracos, mas também cria uma forma de renderização de "arestas vivas". A colocação de vários pequenos *surfels* na região da interseção permite esta correta renderização das “arestas vivas”. Desta forma o problema das "arestas vivas" é resolvido na modelagem. (Figura 3c).

6. Melhorias Propostas e Resultados

Para este trabalho, usamos o método de Dutré e propomos as melhorias: fazemos a classificação da *octree* durante a sua criação; a classificação da *octree* foi otimizada classificando-se os cantos dos seus nós como dentro ou fora do modelo.

Este algoritmo foi implementado na forma de um *plugin* para o software PointShop 3D. O *plugin* foi desenvolvido na linguagem C e no sistema operacional Windows. Os testes foram feitos em um PC x86 *Family* 6, 261600 KB de RAM, sem placa gráfica; sistema operacional Windows 2000. A Figura 4 mostra o resultado das operações de união, interseção e diferença. A Tabela 1 mostra os tempos de processamento.

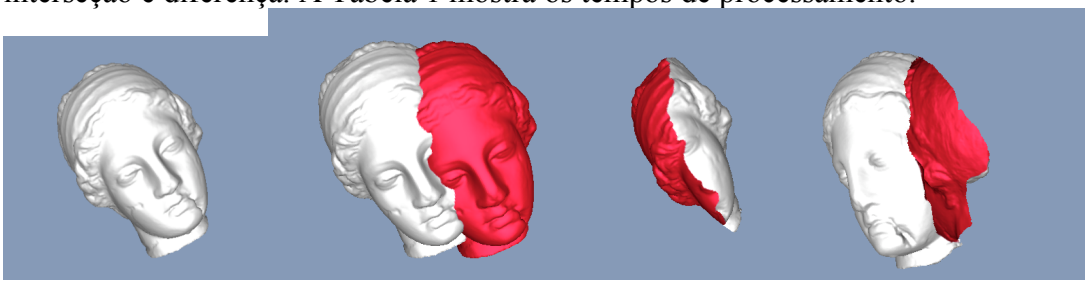

Figura 4-Modelo do PointShop Igea e resultado da união, interseção e diferença.

Operação	Criação das Octrees (seg)	Classificação (seg)	Criação do resultado (seg)	Total (seg)
União	3,9375	1,324	1,8925	7,154
Interseção	3,8585	1,585	0,9547	6,3982
Diferença	3,99875	1,504	1,729	7,29475

Tabela 1- Média dos tempos de processamento para as operações com duas Igeas

7.Comparação entre os dois métodos e conclusão

Em termos de tempo de processamento, comparamos o *plugin* de Pauly com o nosso *plugin* e verificamos empiricamente que os dois métodos apresentam tempos bastante similares. Entretanto, o tempo de processamento do algoritmo aqui implementado ainda pode ser melhorado com o uso do método TINN (que foi proposto em Adams) e com uma otimização no uso de octrees (Sara F. Frisken citado em [1]).

Em termos conceituais, vamos fazer a comparação do uso do que chamamos aqui de Regra 2, do tratamento de interseção, e do tratamento de "arestas vivas". Aparentemente os dois algoritmos são bastante parecidos, um usa octrees e o outro usa *kd-tree*. Mas existe uma diferença conceitual entre dois que merece ser destacada. O algoritmo aqui implementado (método de Dutré) usa a Regra 2 para classificação dos nós de uma *octree* e armazena esta *octree* classificada. Assim na maioria dos casos não será necessário encontrar o *surfel* mais próximo para classficação. O uso desta *octree* classificada levanta uma questão que não foi citada no trabalho de Adams e Dutré que é a proximidade desta estrutura (armazenada) de *octree* classificada com uma representação volumétrica do objeto. Sendo assim, esta estrutura pode ser aproveitada para outras funções além da classificação dos *surfels*. Por exemplo, podemos ter uma aproximação do volume de uma cena, somando os volumes aproximados dos modelos que a compõem.

Outra importante diferença entre os dois trabalhos é que o método de Pauly aplica-se a modelos sólidos ou não sólidos, enquanto que o método de Adams e Dutré aplica-se somente a modelos sólidos.

Quanto ao tratamento de interseção, em termos de visualização todos os dois resolvem bem o problema. Entretanto, o tratamento de Adams e Dutré não depende da renderização sendo, portanto, mais portável. Além disso, uma desvantagem do método de Pauly é o uso da representação implícita MLS do objeto para obter os pontos exatamente na curva de interseção, o que torna o tratamento de interseção mais complexo e não traz vantagem significativa para este tratamento de interseção.

Concluímos então, que o método de Adams e Dutré apresenta as seguintes vantagens em relação ao de Pauly: a estrutura de octree classificada, além de ser usada para classificação, fornece informações adicionais como volume aproximado do objeto; o tempo de processamento no método de Adams e Dutré tende a ser melhor que no método de Pauly, se implementadas as melhorias citadas e se colocássemos em consideração o uso da representação implícita MLS; o tratamento de interseção de Adams e Dutré apresenta a vantagem da portabilidade, ou seja, irá funcionar independente do algoritmo de renderização utilizado. E as vantagens do método de Pauly em relação ao de Adams e Dutré: a colocação de surfels exatamente na curva de interseção e a possibilidade de trabalhar com modelos sólidos ou não sólidos.

Referências

[1] LEAL, HELOISA REIS (2004). Operações Booleanas em Modelagem por Pontos. **Dissertação de Mestrado**, Departamento de Informática/PUCRio.

[2] ADAMS, B.; DUTRÉ, P. Interactive Boolean Operations on Surfel-Bounded Solids. **Proceedings of SIGGRAPH 2003**, San Diego, USA, 2003.

[3] PAULY, M.; KEISER, R.; KOBBELT, L.; GROSS, M. Shape Modeling with Point-Sampled Geometry. **SIGGRAPH 2003**, Computer Graphics Proceedings, Annual Conference Series. ACM Press / ACM SIGGRAPH, 2003.