UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Revisitando o Problema de Visibilidade para Visualização Tridimensional

Ícaro Lins Leitão da Cunha

Orientador: Prof. Dr. Luiz Marcos Garcia Gonçalves

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e de Computação da UFRN (área de concentração: Engenharia de Computação) como parte dos requisitos para obtenção do título de Doutor em Ciências em Engenharia Elétrica e de Computação.

Número de ordem PPgEE: D105
Natal, RN, janeiro de 2014

Catalogação na publicação

Thiago Ferreira Cabral de Oliveira - CRB/15 - 628.
Bibliotecário Documentalista IFPB Campus - Cajazeiras.

C972r CUNHA, Ícaro Lins Leitão da Cunha

Revisitando o Problema de Visibilidade para Visualização Tridimensional / Ícaro Lins Leitão da Cunha; -Natal, 2014.

Orientador: Luiz Marcos Garcia Gonçalves

Tese (doutorado) - Universidade Federal do Rio Grande do Norte. Centro de Tecnologia. Programa de Pós-Graduação em Engenharia Elétrica.

1. Visualização. 2. 3D 3. Renderização. I. Título III. Assunto.

CDU 004.383(5)

Revisitando o Prolema de Visibilidade para Visualização Tridimensional

Ícaro Lins Leitão da Cunha

Tese de Doutorado aprovada em 22 de janeiro de 2014 pela banca examinadora composta pelos seguintes membros:

Prof. Dr. Luiz Marcos Garcia Gonçalves (orientador) ..... UFRN

Prof. Dr. Aquiles Medeiros Filgueira Burlamaqui (Examinador Interno) UFRN

Prof. Dr. Rummenigge Rudson Dantas (Examinador Externo ao PPgEEC) .... UFRN

Prof. Dr. Ricardo Cordeiro Farias (Examinador Externo) ..... UFRJ

Prof. Dr. Anderson Abner Santana de Souza (Examinador Externo) .... UERN

Aos meus avós, por todo apoio e exemplo de vida.

Agradecimentos

Agradeço primeiramente a Deus, pois sem Ele esta jornada não seria cumprida;

À minha noiva Rafaela Maria Braga de Farias pela paciência e apoio durante toda esta jornada;

Aos meus pais, que me serviram de grande exemplo de vida;

A minha sogra que torceu pela minha vitória mas que infelizmente não está mais aqui para comemorar mais essa conquista;

À minha família, em especial, minhas irmãs e meus avôs;

Ao meu orientador, Luiz Marcos Garcia Gonçalves, sou grato pela sua orientação e dedicação;

Ao professores Ricardo Farias, Amitabh Varshney e Marcelo Siqueira, pelas suas contribuições e colaboração;

Aos colegas do Laboratório NatalNet, pelas sugestões e pela amizade durante o meu doutorado. Além de também agradecer aos colegas do laboratório GVIL da Universidade de Maryland, em especial a Sujal Bista;

Aos demais colegas de pós-graduação, pelas críticas e sugestões;

À CAPES, pelo apoio financeiro, através do Projeto RH-TVD. E ao CNPq pelo financiamento do meu doutorado sanduíche.

Resumo

Nós revisitamos o problema de visibilidade, que visa determinar um conjunto de primitivas potencialmente visíveis em um conjunto de dados geométricos representados por uma estrutura de dados, por exemplo uma malha de polígonos ou de triângulos, propondo uma solução para acelerar o processamento em aplicações em visualização tridimensional. Introduzimos uma estrutura enxuta, no sentido de abstração e redução de dados, que pode ser usada para aplicações online e interativas. O problema de visibilidade é especialmente importante na visualização 3D de cenas representadas por grande volume de dados, em que não é interessante manter todos os polígonos da cena em memória. Isso implicaria em um maior tempo gasto na renderização, ou sendo até mesmo impossível mantê-los todos em volumes imensos de dados. Nestes casos, dada uma posição e uma direção de visualização, o objetivo principal é determinar e carregar o mínimo possível de primitivas (polígonos) da cena, visando acelerar a etapa de renderização. Para este propósito, nosso algoritmo executa o corte de primitivas (*culling*) usando um paradigma híbrido baseado em três modelos conhecidos. A cena é subdividida em células de uma grade, sendo associada a cada uma dessas células as primitivas pertencentes a elas, e finalmente determinado o conjunto de primitivas potencialmente visíveis. A novidade é o uso da triangulação J_1^a para criar a subdivisão em grade. Escolhemos esta estrutura devido às suas características relevantes de adaptatividade e algebrismo (facilidade de cálculos). Os resultados mostram uma melhoria substancial sobre os métodos tradicionais quando aplicados separadamente. O método introduzido neste trabalho pode ser usado em dispositivos sem processador dedicado ou com baixo poder de processamento, e ainda, pode ser utilizado para visualizar dados através da Internet, tal como em aplicações de museus virtuais.

Palavras-chave: Visibilidade; Estrutura de Visualização; Visualização 3D em Tempo Real; Corte de Primitivas Ocultas

Abstract

We revisit the problem of visibility, which is to determine a set of primitives potentially visible in a set of geometry data represented by a data structure, such as a mesh of polygons or triangles, we propose a solution for speeding up the three-dimensional visualization processing in applications. We introduce a lean structure , in the sense of data abstraction and reduction, which can be used for online and interactive applications. The visibility problem is especially important in 3D visualization of scenes represented by large volumes of data, when it is not worthwhile keeping all polygons of the scene in memory. This implies a greater time spent in the rendering, or is even impossible to keep them all in huge volumes of data. In these cases, given a position and a direction of view, the main objective is to determine and load a minimum ammount of primitives (polygons) in the scene, to accelerate the rendering step. For this purpose, our algorithm performs cutting primitives (culling) using a hybrid paradigm based on three known techniques. The scene is divided into a cell grid, for each cell we associate the primitives that belong to them, and finally determined the set of primitives potentially visible. The novelty is the use of triangulation J_1^a to create the subdivision grid. We chose this structure because of its relevant characteristics of adaptivity and algebrism (ease of calculations). The results show a substantial improvement over traditional methods when applied separately. The method introduced in this work can be used in devices with low or no dedicated processing power CPU, and also can be used to view data via the Internet, such as virtual museums applications.

Keywords: Visibility Culling; Visualization Structure; Real-time 3D Visualization; Hidden Primitive Culling.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Em gráficos 3D, a determinação da superfície oculta (também conhecido como remoção da superfície oculta, remoção de oclusão ou determinação de superfícies visíveis) é o processo usado para determinar que parte das superfícies de uma cena são visíveis a partir de um ponto de vista e, por conseguinte, quais os que não são visíveis [Foley et al. 1990]. Um algoritmo para a determinação da superfície oculta é uma solução para o problema da visibilidade, o que é um dos principais problemas nos primórdios da Computação Gráfica 3D [Cohen-Or et al. 2003]. O processo de determinação da superfície oculta é às vezes chamado determinação de oclusão. O análogo para linha é a remoção de linha oculta. Determinar a superfície oculta é necessário para processar uma imagem corretamente, de modo que não se pode olhar através de (ou atravessar) paredes em aplicações de realidade virtual, por exemplo. Existem muitos trabalhos na literatura que tratam desse problema para aplicações interativas, o algoritmo preferencial para resolver o problema de remoção de superfície escondida é o *Z-buffer* [Catmull 1974].

Neste trabalho, revisitamos o problema de visibilidade, ou seja, o problema de computar um conjunto de primitivas potencialmente visível em um modelo de cena, tendo em vista aplicações interativas e de tempo real. À luz do presente estudo, nós propomos melhorias no modelo da *pipeline* de visualização utilizando uma estrutura de visualização que minimiza a quantidade de dados inúteis a serem carregados para gerar a visualização de uma determinada cena. A fim de realizarmos isto, dividimos a cena em uma grade 3D; em seguida, associamos a cada bloco da grade suas respectivas primitivas que lhe pertencem e, finalmente, determinamos o conjunto de primitivas visíveis a partir do ponto vista que observa a cena. Uma de nossas contribuições é que o nosso método utiliza a estrutura base da Triangulação J_1^a proposta por [Castelo et al. 2006] para criar a grade de subdivisão. Nós escolhemos esta estrutura devido à suas operações algébricas e capacidade adaptiva.

Além de revisitarmos o problema de visibilidade, também trataremos o problema de visualização de dados médicos. Quanto a visualização de dados médicos, sua visualização 3D tem ganhado cada vez mais importância, especialmente na área de visualização de imagens da estrutura cerebral.

Tendo em vista que uma aplicação de visualização de dados médicos também requer uma quantidade grande de informações, propomos também utilizar a estrutura descrita acima na implementação de um visualizador de dados cerebrais. Neste caso, a aplicação

será voltada a visualizar dados cerebrais obtidos a partir de exame de Imageamento por Difusão por Curtose (*Diffusion Kurtosis Imaging* - DKI).

1.1 Contextualização e Motivação

Além dos avanços na tecnologia de hardware e software realizados nas últimas décadas, a determinação de visibilidade ainda é um dos principais problemas na computação gráfica. Vários algoritmos capazes de remover superfícies ocultas têm sido desenvolvidos ao longo dos anos para resolver este problema [Cohen-Or et al. 2003]. Esses algoritmos determinam qual o conjunto de primitivas que compõem uma cena em particular é visível a partir de uma dada posição de visualização. Em [Cohen-Or et al. 2003] ficou estabelecido que o problema básico é em grande parte resolvido, no entanto, sabe-se que, devido à constante demanda por maior quantidade de dados 3D (maior realismo em cenas), algoritmos como *Z-buffer* e outras abordagens clássicas podem ter problemas para garantir a visualização da cena em tempo real. Na verdade, concordando com Cohen et al., é possível encontrar trabalhos na literatura como [Jones 1971], [Clark 1976] and [Meagher 1982] que abordaram esta questão nos primórdios da Computação Gráfica. No entanto, lembramos que se deve dar atenção para a importância real deste problema durante a evolução da CG. Pesquisadores como [Airey et al. 1990], [Teller & Séquin 1991], [Teller 1992] e [Greene et al. 1993], centraram-se sobre esta questão e revisitaram o problema de visibilidade para acelerar a visualização.

Escolhemos revisitar o problema de visibilidade de cenas 3D, pois não encontramos na literatura nenhuma publicação recente. E como as aplicações de navegação on-line e em tempo real em mundos virtuais (museu, edifícios e saúde) exigem quantidade significativa de primitivas a serem renderizados em tempo real, decidimos revisitar esse problema de visibilidade para testar se ainda podem ser melhoradas. De fato, observa-se que as técnicas anteriores tem algumas dificuldades como a complexidade da remoção de oclusão para analisar toda a cena ou a quantidade de cálculos necessários para determinar para onde as primitivas estão voltadas ou se estão contidos no volume de visualização (frustum), em métodos de descarte de visibilidade. Estes casos não são tão triviais, principalmente com quantidade enorme de dados ou quando a cena é modelada de uma vez. Por exemplo, isso pode ocorrer em uma sala de museu virtual com muitas esculturas (ilustrado na Figura 1.1 um dos museus virtuais desenvolvido pelo sistema GTMV do laboratório NatalNet [Dantas et al. 2009]) ou em cenas externas de uma cidade.

Superamos essas dificuldades, propondo uma abordagem diferente de trabalhos anteriores e adaptando as técnicas simples de remoção de primitivas para nosso contexto. Utilizamos uma estrutura de visualização que é capaz de obter um conjunto de primitivas potencialmente visível da cena a partir do ponto de vista de um utilizador. Como dito acima, esta estrutura é construída ao subdividir toda a cena por uma grade baseada na estrutura da Triangulação J_1^a . Com isso, conseguimos melhorias significativas nos resultados. Em particular, nosso método pode lidar com dados complexos, reduzindo os requisitos de recursos.

Quanto a visualização de modelos cerebrais desejamos auxiliar a identificação do comportamento celebral usando dados obtidos por DKI em casos onde o simples uso

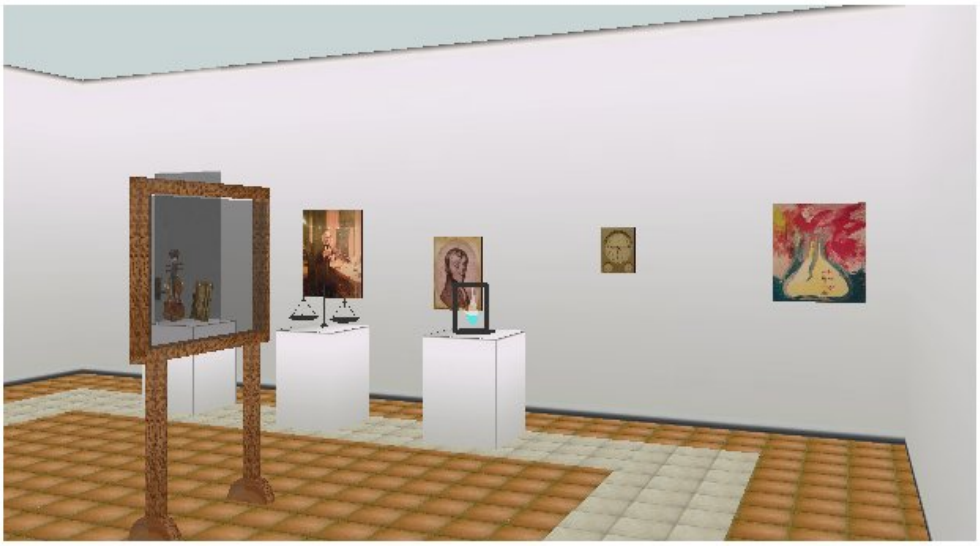


Figura 1.1: Representação virtual do Museu do NAC localizado na Universidade Federal do Rio Grande do Norte. Essa aplicação baseada em web foi desenvolvida pelo sistema GTMV.

de dados DTI (Imageamento por Difusão Tensorial) não funciona completamente [Jensen et al. 2005]. Mais especificamente, buscamos visualizar a estrutura cerebral ao longo do seu processo de cura. Para isso, reconstruimos a estrutura cerebral a partir dos dados obtidos e utilizamos a nossa estrutura para auxiliar em sua visualização.

1.1.1 Aplicações

A fim de testarmos a estrutura de visualização, além de verificarmos sua eficiência em experimentos de desempenho, propomos 3 aplicações:

- **Visualização com Percepção 3D** - Aplicação em um programa de visualização 3D onde o ponto de vista é deslocado (transladado ou rotacionado). Esse deslocamento gera uma sensação de percepção 3D da cena e é baseado no trabalho proposto por [Bista et al. 2013];
- **Caverna Virtual** - Aplicação da estrutura em uma simulação do seu uso em uma caverna virtual. Nesta aplicação, a cena é observada por mais de um ponto de vista;
- **Visualização de Dados Médicos** - Aplicação de visualização de estruturas cerebrais descrita nas sessões anteriores.

1.2 Objetivos e Contribuições

Contribuições: A principal contribuição deste trabalho é a própria estrutura de visualização que é capaz de determinar a oclusão de duas maneiras: oclusão interna do bloco e oclusão entre blocos adjacentes. O uso de funções algébricas para acesso rápido dos

blocos visíveis, faces desses blocos e determinação de adjacência entre blocos também é outra contribuição que o reforço do processo de visualização ajudou.

Utilizamos a estrutura de visualização para nova forma de visualizar dados celebrais obtidos a partir exames feitos com Imageamento de Difusão Catódica (*Difusion Kurtosis Imaging*). Esses dados permitem a geração de malhas 3D detalhadas capazes de identificar o comportamento da difusão da água dentro do cérebro humano e assim identificando a estrutura desse cérebro.

1.3 Organização da Tese

Nesta tese, serão detalhados os conceitos estudados e as atividades realizadas. Ela está organizada da seguinte forma:

- Capítulo 2: neste capítulo, são abordados os conceitos ditos necessários para o desenvolvimento da tese. Daremos atenção especialmente no
- Capítulo 3: neste capítulo, apresentamos uma revisão bibliográfica e maior detalhamento da contextualização do trabalho apresentado. Abordaremos também os trabalhos relevantes quanto ao imageamento de estruturas cerebrais;
- Capítulo 4: neste capítulo, formulamos os problemas que abordamos nessa tese e propomos suas soluções;
- Capítulo 5: neste capítulo, detalhamos as implementações das soluções propostas no Capítulo 4 através de fluxograma e visão sistemática de cada implementação;
- Capítulo 6: neste capítulo, apresentamos os resultados da análise de desempenho da estrutura de visualização. Apresentamos também aplicações onde utilizamos essa estrutura, entre elas, a aplicação de visualização de dados cerebrais;
- Capítulo 7: neste capítulo, é feita uma apreciação sobre o trabalho desenvolvido e serão detalhados alguns trabalhos futuros.

Capítulo 2

Embasamento Teórico

Neste capítulo, revisamos alguns conceitos básicos quanto a visualização de dados 3D. Introduzimos noções de oclusão 3D, modelagem geométrica, estrutura de dados e codificação de dados 3D. Além disso, apresentamos conceitos de imagens médicas com foco em Imageamento por Difusão Curtótica.

2.1 Conceitos Iniciais

Apresentamos aqui as definições de conceitos utilizados ao longo deste capítulo. Estes conceitos foram obtidos do trabalho de [Castelo 1992]. Começamos pela definição de Espaço Afim:

Definição 1 (Espaço Afim) *Dados* $v_0, v_1, \ldots, v_n \in \mathbb{R}^m$*, o espaço*

$$S = aff\{v_0, \ldots, v_n\} = \left\{ v \in \mathbb{R}^m : \sum_{i=0}^{n} \lambda_i v_i = v, e \sum_{i=0}^{n} \lambda_i = 1 \right\}$$

é denominado espaço afim gerado por $v_0, v_1, \ldots, v_n$*.*

Definição 2 (Célula) *Uma célula convexa afim gerada pelos pontos* v_0*,* v_1*, ..,* v_n *é definida como sendo o conjunto:*

$$\sigma = [v_0, v_1, \ldots, v_n] = \left\{ v \in \mathbb{R}^m : \sum_{i=0}^{n} \lambda_i v_i = v, \sum_{i=0}^{n} \lambda_i = 1 \ e \ \lambda_i \geq 0 \right\}.$$

Em outra palavras, σ é o menor conjunto convexo contendo os pontos v_0, ..., v_n.

Definição 3 (Dimensão de uma Célula) *A dimensão de uma célula convexa afim* $\sigma = [v_0, v_1, \ldots, v_n]$ *é definida como sendo a dimensão do espaço afim, ou seja, o menor número de vetores linearmente independentes entre* $v_1 - v_0, v_2 - v_0, \ldots, v_n - v_0$*.*

Um caso particular de célula que é comumente tratado na literatura é o de simplexo.

Definição 4 (Simplexo) *Um simplexo de dimensão d (d-simplexo) é uma célula convexa afim de dimensão d gerada por* $d+1$ *pontos.*

Por convenção, o 0-simplexo é chamado de vértice, o 1-simplexo de aresta, o 2-simplexo de triângulo e o 3-simplexo de tetraedro.

Definição 5 (Bordo de uma Célula) *O bordo de uma célula σ, denotado por $\partial(\sigma)$, é o conjunto de sub-células satisfazendo:*

1. *Se $\sigma' \in \partial(\sigma)$, então σ' é gerado por um subconjunto de vértices de σ;*
2. *$\forall p \in \sigma'$, onde $\sigma' \in \partial(\sigma)$, não existe uma bola contendo p em $aff(\sigma)$ inteiramente contida em σ';*
3. *Se σ'_1 e σ'_2 pertencem a $\partial(\sigma)$ e tem a mesma dimensão $\Rightarrow$ $\sigma'_1 \nsubseteq \sigma'_2$ e $\sigma'_2 \nsubseteq \sigma'_1$.*

Definição 6 (Face) *Sejam $\gamma = [w_0, \ldots, w_n]$ uma p-célula convexa afim e $\sigma = [v_0, \ldots, v_m]$ uma q-célula. Então, se $\{w_0, \ldots, w_n\} \subset \{v_0, \ldots, v_m\}$, diz-se que γ é uma face do bordo de σ ($\partial(\sigma)$) se:*

1. *$\gamma \subset \partial(\sigma)$;*
2. *Se existe alguma outra sub-célula η cujo espaço afim é o mesmo que o de γ (ou seja, $aff(\gamma) = aff(\eta)$), então $\eta \subset \gamma$.*

Definição 7 (Decomposição Celular) *Uma coleção C de células convexas afins é dita uma decomposição celular de um conjunto $K \subset \mathbb{R}^m$ se:*

1. *$K = \cup_{\sigma \in C} \sigma$;*
2. *Se σ_1, $\sigma_2 \in C$ então $\sigma_1 \cap \sigma_2 = \emptyset$ ou $\sigma_1 \cap \sigma_2 \in C$;*
3. *Todo subconjunto compacto de K intercepta um número finito de células de C.*

Definição 8 (Triangulação) *Uma decomposição celular C de $K \subset \mathbb{R}^m$ é chamada de triangulação de K, se todas as células de C são simplexos.*

Definição 9 (Complexo celular) *Um complexo celular C é um conjunto finito de células convexas afins que satisfazem:*

1. *se $\sigma \in C$ e γ é face de σ então $\gamma \in C$;*
2. *se σ_1 e $\sigma_2 \in C$ então $\sigma_1 \cap \sigma_2 = \emptyset$ ou $\sigma_1 \cap \sigma_2$ é uma face comum de σ_1 e σ_2.*

A condição 2 da Definição 9 garante que a interseção entre duas células quaisquer se dá somente por uma face em comum entre eles, impedindo assim interseções indesejadas. A partir daqui, todos os complexos celulares abordado nessa tese serão chamadaos de **malhas**.

Neste trabalho, o conceito de adjacência ou conectividade entre elementos será importante para entender a operação de transição entre elementos/blocos vizinhos.

Definição 10 (Células Adjacentes) *Uma célula σ_1 é dita adjacente (vizinha) a outra célula σ_2 se $\sigma_1 \cap \sigma_2$ é uma face comum de σ_1 e σ_2.*

Além desse conceitos básicos, definimos também:

Definição 11 (Codificação Geométrica) *A codificação geométrica de uma malha é a sua representação numérica em forma de lista de seus elementos que a compõem. Neste caso, um vértice deve ser representado no mínimo pelas suas coordenadas e uma face deve ser representado pelos índices dos vértices pertencentes a ela. Exemplo de codificação geométrica: lista de vértices e lista de faces.*

Definição 12 (Codificação por Conectividade) *A codificação por conectividade é a representação numérica da conectividade/adjacência desejada entre células que compõem uma malha. Neste caso, se uma célula é dita adjacente a outra, ambas podem se referenciar através da codificação por conectividade.*

2.2 Visualização de Dados 3D

Visualização de objetos 3D é um processo relativamente mais complexo que o processo de visualização 2D. No caso de 2D, somente é necessário especificar a janela de visualização no mundo 2D e um ponto de vista (*viewport*) na superfície de visualização 2D. Conceitualmente, os objetos da cena são recortados pela janela e transformados dentro do *viewport* para visualização. A complexidade de visualização 3D é causada, em parte, pela adição de mais uma dimensão e também, em parte, pelo fato dos dispositivos de visualização serem somente em 2D [Foley et al. 1990].

Operações de projeções são então utilizadas para resolver o problema de como visualizar uma cena 3D em um dispositivo 2D. Basicamente, objetos 3D são projetados em um plano de projeção 2D.

Em visualização 3D, especifica-se o volume de visualização do mundo (**frustum**), o tipo de projeção do mundo em um plano de projeção e uma *viewport* na superfície de visualização. Conceitualmente, os objetos da cena 3D são recortados pelo volume de visualização 3D e então são projetados. O conteúdo projetado no plano são então transformados em uma *viewport* para visualização. Antes de todas essas operações serem realizadas, deve se levar em conta que cada objeto 3D que compõem uma cena tem seu próprio sistema de coordenadas. Devido a isso esses objetos são transformados para o sistema de coordenadas do mundo, essa transformação se chama transformada de modelagem. A figura 2.1 ilustra uma visão geral do processo de visualização da cena 3D.

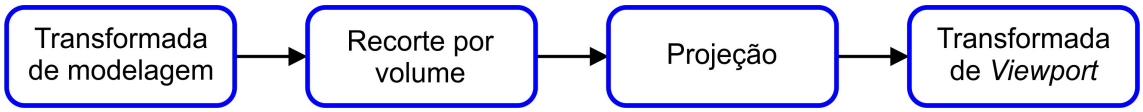


Figura 2.1: Visão geral do processo de visualização de uma cena 3D.

Além da operação de recorte, durante o processo de visualização de uma cena 3D são executadas outras operações que eliminam elementos da cena/objeto que não são visíveis. E, muitas vezes, cenas 3D são compostas por uma quantidade grande de objetos 3D e esses por si só podem ser complexos a fim de proporcionar realismo da cena. Devido à uma alta quantidade de dados essas operações de detecção de visibilidade podem ser bastante custosas para a execução de visualização em tempo real.

2.2.1 Oclusão de Dados 3D

Em aplicações de visualização de dados 3D sempre haverá a possibilidade de que nem toda a cena esteja sendo exibida a partir de um determinado ponto de vista. Neste caso, se uma ou mais primitivas gráficas que não estão sendo visualizadas forem processadas durante a visualização, haverá um gasto de processamento desnecessário e muitas vezes poderá reduzir a taxa de exibição drasticamente. Para isso, se faz necessário determinar quais primitivas estão sendo utilizadas para gerar a visualização da cena.

O problema de determinação de visibilidade visa remover rapidamente as primitivas que não contribuem para a geração da imagem final da cena. Este passo é realizado antes do passo de remoção da superfície oculta, e somente o conjunto de primitivas que contribuem para, pelo menos, um pixel na tela deve ser processado. A determinação de visibilidade é executada usando as seguintes estratégias: *Back-Face* e *View-Frustum culling* [Foley et al. 1990]. *Back-Face culling* como o nome diz em inglês utiliza apenas as primitivas voltadas para a direção da câmera, e *View-Frustum culling* visa utilizar somente as primitivas dentro do frustum de visualização. Ao longo dos anos, técnicas hierárquicas eficientes foram desenvolvidas [Clark 1976], [Garlick et al. 1990], [Kumar et al. 1996], bem como outras otimizações [Assarsson & Möller 2000], [Slater & Chrysanthou 1997], a fim de acelerar a determinação de visibilidade. Além dessas técnicas, a técnica de remoção de oclusão (*Occlusion Culling*) evita processar primitivas que estão escondidas (oclusas) por outras primitivas na cena. Esta técnica é mais complexa, pois envolve uma análise da relação global entre todas as primitivas da cena. Figura 2.2 ilustra a relação entre as três técnicas de *culling* conhecidas.

2.2.2 Visualização 3D baseada em imagens

Outras técnicas para visualização de cena 3D foram propostas. Entre elas, as técnicas baseadas em imagem utilizam conjuntos de imagens para gerar não só modelos 3D, como também, visualizar a cena por outros pontos de vista.

Mesmo que esses técnicas se baseiam em uma abordagem diferente da abordagem tomada neste trabalho. Acreditamos que valha a pena discutir suas vantagens e desvantagens.

Existem diversas técnicas baseadas em imagem, sites como o Stockholm 360 [1] utilizam fotos panorâmicas (como a foto do Jardim Bagh-e Eram ilustrada na Figura 2.3) para gerar uma visão em 360 graus para criar um tour virtual de museus e prédios. A Figura 2.4 ilustra a visualização de uma das galerias do Museu do Louvre de Paris.

Para aplicações, como as de museus virtuais, essas técnicas são capazes de representar (com boa qualidade) galerias/ambientes com uma quantidade relativamente menor de dados, em comparação a visuaização de uma cena 3D com objetos gráficos mais texturas. Porém, deve-se tomar muito cuidado para não ocluir e não gerar distorção dos objetos fotografados. Além disso essa técnica limita a capacidade de interatividade do usuário para navegar ao longo da cena.

[1] Stockholm 360, site: http://www.stockholm360.net/ - última vez acessado em 12/12/2013.

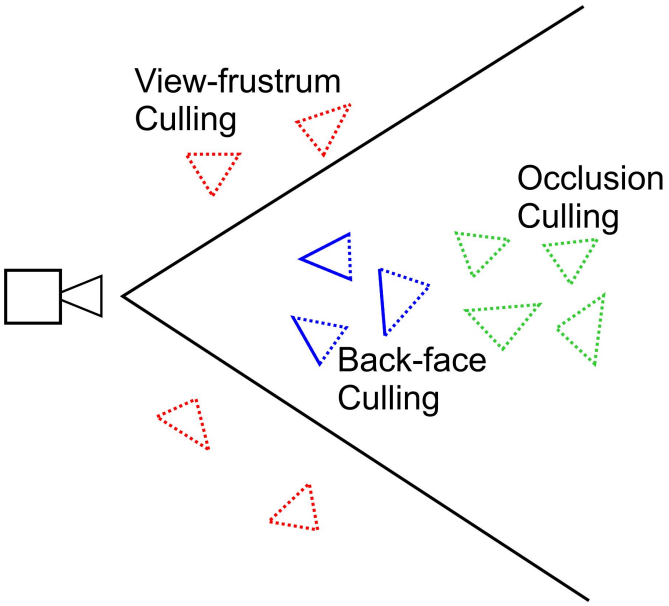


Figura 2.2: Tipos técnicas de culling: *view-frustum culling* (em vermelho), *back-face culling* (em azul), e *culling* por oclusão (em verde). Neste caso, somente as arestas representadas por retas contínuas estão visíveis.

2.3 Modelagem Geométrica

Modelagem geométrica é um ramo da matemática aplicada e geometria computacional que trata do problema de geração, manipulação geométrica e topológica dos objetos graficos no computador. Nela, são estudados os métodos e algoritmos para a descrição matemática de formas geométricas que formam os objetos gráficos (genericamente chamados de modelos) [Gomes & Velho 1990].

As formas estudadas nesta área são principalmente de duas ou três dimensões, embora muitas de suas ferramentas e princípios podem ser aplicados a conjuntos de qualquer dimensão finita. Hoje em dia, a maior parte da modelagem geométrica é feita com computadores e por aplicações baseadas em computador. Modelos bidimensionais são importantes em tipografia computacional e desenho técnico. Modelos tridimensionais são fundamentais em desenho auxiliado por computador (*computer-aided design*) e manufatura (CAD/CAM), e amplamente utilizado em muitas áreas técnicas, tais como engenharia civil e mecânica, arquitetura, geologia, processamento de imagens médicas [Foley et al. 1990] etc.

Modelos geométricos são geralmente distinguidos de modelos processuais e orientados a objetos, que definem implicitamente a forma por um algoritmo opaco que gera a sua aparência. Eles também se contrastam com as imagens digitais e modelos volumétricos que representam a forma como um subconjunto de uma partição fina regular do espaço; e com um modelo fractal que dá uma definição infinitamente recursiva da forma. No en-

Figura 2.3: Foto panorâmica em 360 graus do Jardim Bagh-e Eram (Jardim do Éden) utilizada pelo site Stockholm 360.

Figura 2.4: Visualização de uma cena galeria de Michelangelo do Museu do Louvre do site Stockholm 360.

tanto, estas distinções são freqüentemente imprecisas: por exemplo, uma imagem digital pode ser interpretada como um conjunto de quadrados coloridos; e formas geométricas, como círculos, são definidas por equações matemáticas implícitas. Além disso, um modelo fractal produz um modelo paramétrico ou implícito, quando a sua definição recursiva é truncada para uma profundidade finita.

Objetos gráficos complexos são cada vez mais utilizados em várias aplicações, incluindo em aplicações de jogos, em design na engenharia, na navegação de ambientes de arquitetura, na realidade virtual e na visualização científica. Entre as várias maneiras de representação, malhas triangulares proporcionam um meio eficaz para representar modelos de malha 3D. Normalmente, conectividade, geometria, e propriedade dos dados são usados para representar uma malha poligonal 3D. Conectividade descrever a relação de adjacência entre vértices; dados de geometria especifica a localização de cada vértices; e dados de propriedade especificar vários atributos, como o vetor normal, material de reflexão, e coordenadas de textura. As informações geométricas e de propriedade geralmente são atribuidas a todos os vértices.

A tendência atual é que aplicações gráficas demandam cada vez mais um maior realismo de objetos e cenas gráficas, este fato é evidente quando observamos jogos digitais cada vez mais reais e o crescimento atual de aplicações de realidade virtual que simulam

rotinas do mundo real. Para atingir um alto nível de realismo, são necessários modelos cada vez mais complexos, e eles são obtidos de várias fontes, tais como software de modelagem e digitalização 3D. Eles exigem geralmente uma grande quantidade de espaço de armazenamento e/ou a largura de banda de transmissão, no formato de dados bruto. A medida que o número e complexidade de malhas 3D aumentam exponencialmente, demandas de recursos mais elevados são pedidas quanto ao espaço de memória, poder de processamento e largura de banda da internet. Entre estes aplicativos, a largura de banda da rede é o gargalo mais grave em aplicações gráficas baseadas em rede que exigem interatividade em tempo real entre múltiplos usuários. Assim, se torna essencial uma redução eficiente de dados gráficos sendo processados.

No escopo do nosso trabalho, usaremos somente malhas triangulares, porém o nosso método pode ser facilmente extendido para o uso em cenas compostas por outros tipos de malhas superficiais. Por isso, a fim de generalizar os tipos de malhas utilizadas ao longo dessa tese, chamaremos os elementos que as compõem de **primitivas**.

2.3.1 Triangulações Retangulares

As triangulações J1 e K1 resultam de subdivisões de malhas quadradas regulares onde os quadrados da malha são subdivididos por uma de suas diagonais [Gonçalves 1996] [Gonçalves 1997]. Na K1, tomam-se as diagonais paralelas entre si e na J1 alternadas (como na bandeira do Reino Unido). A Figura 2.5 ilustra a diferença entre essas triangulações. Nessas triangulações regulares, todos os triângulos são retângulo-isósceles. As triangulações retangulares adaptativas (RA), generalizações dessas, são aquelas que envolvem apenas triângulos retângulo-isósceles (em qualquer arranjo e de quaisquer tamanhos).

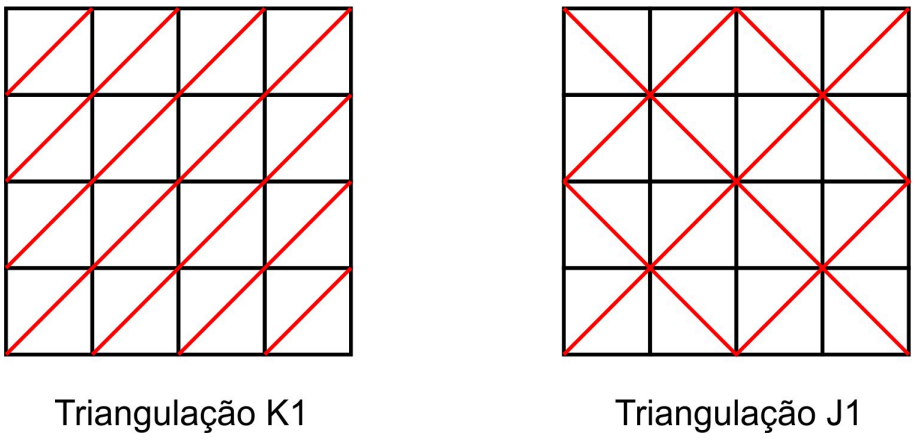


Figura 2.5: Ilustração das triangulações K1 e J1. A diferença básica entre essas triangulações é o fato que as diagonais (em vermelho) são paralelas em K1 e alternadas em J1.

Uma vasta família de triangulações RA (a que nos restringiremos aqui) engloba aquelas que podem ser obtidas a partir de triangulações J1 ou K1 por sucessivas operações bá-

sicas de subdivisão por refinamento. O refinamento resulta da subdivisão de triângulos em dois cortando-os através de um segmento unindo o ponto médio de suas hipotenusas aos seus respectivos vértices opostos (o que mantem a natureza geométrica dos triângulos). Para preservar a estrutura de triangulação, o refinamento de um triângulo requer em geral a subdivisão (simultânea) de outros. No entanto, é possível obter-se qualquer triangulação RA (refinada) apenas por sucessivos refinamentos de núcleos [Persiano et al. 1993] de triangulações J1 ou K1 como ilustra-se a seguir.

A Figura 2.6 ilustra uma triangulação RA classificando os seus triângulos de acordo com os conceitos núcleo, fonte e bacia introduzidos em [Persiano et al. 1993] e são apresentados a seguir.

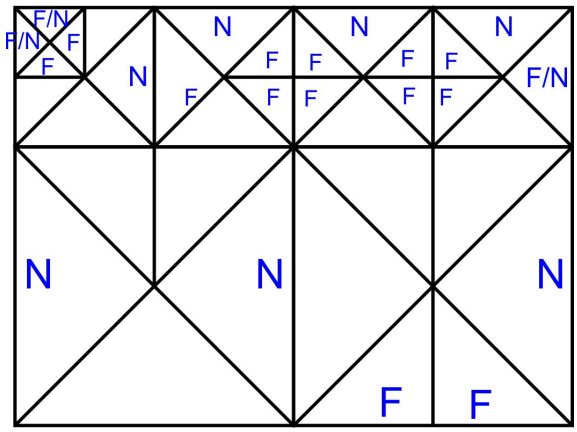


Figura 2.6: Ilustração dos elementos da triangulação RA. F representa a fonte, N o núcleo e F/N a fone e o núcleo.

- Núcleo (N): conjunto formado por dois triângulos com uma hipotenusa comum; donomina-se núcleo (degenerado) também a todo triângulo com sua hipotenusa no bordo da triangulação. O refinamento mais elementar de uma triangulação RA corresponde à subdivisão (ao meio) dos triângulos de um de seus núcleos. Por refinamento de núcleos, produz-se todos os possíveis refinamentos de triangulações RA.
- Caminho de refinamento (C): é uma sequência de triângulos em que cada triângulo é vizinho de seu sucessor por sua hipotenusa; o caminho maximal que se inicia em um triângulo, termina necessariamente em um triângulo pertencente a um núcleo; o seu núcleo terminal. Se um triângulo deve ser subdividido ao meio, como parte de um refinamento, é minimamente necessário subdividir-se ao meio todos os triângulos que o sucedem em seu caminho até um núcleo terminal. A essa sequencia de subdivisões denomina-se refinamento básico do triângulo. Note-se que todo refinamento básico pode ser decomposto em uma sequência de refinamentos de núcleo: refina-se reiteradamente o núcleo terminal do caminho maximal do triângulo, até que ele próprio seja subdividido.

- Fonte (F): todo triângulo cujos catetos não são hipotenusas de seus vizinhos. Esses triângulos são iniciais a qualquer caminho em que ocorrem.
- Bacia de um núcleo: conjunto de todos os triângulos cujos caminhos terminam em um núcleo.

Neves e Persiano [Neves & Persiano 1997] descrevem um esquema iterativo genérico para construir-se triangulações RA minimais segundo critérios especificados. Tipicamente determina-se a triangulação RA refinada minimal que atende aos requisitos de um problema caracterizados por um critério de aprovação fundado nos triângulos: a triangulação satisfaz os requisitos se cada triângulo atende ao critério de aprovação. O método de construção da triangulação RA minimal envolve exclusivamente refinamentos básicos sucessivos de uma triangulação J1 de partida.

A Figura 2.7 ilustra uma exemplo de malha triangular 2D obtida através de uma triangulação RA.

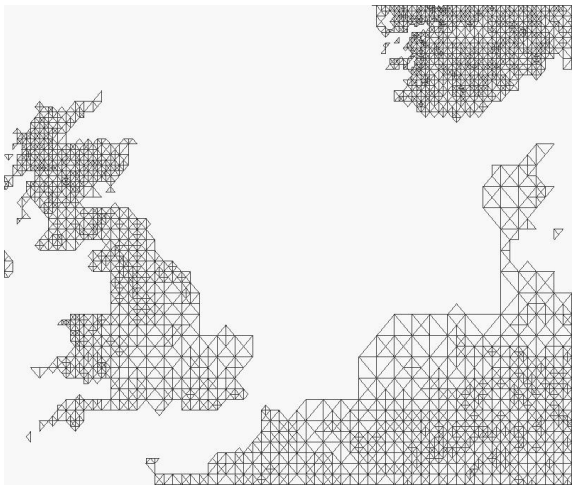


Figura 2.7: Ilustração de uma triangulação 2D obtida usando uma técnica de triangulação retangular adaptativa.

2.3.2 Triangulação J_1^a

Proposta por Castelo et al. [Castelo et al. 2006], a triangulação J_1^a é uma estrutura algebricamente definida que é capaz de ser construída em qualquer dimensão. Para acomodar aspectos locais, a triangulação J_1^a lida com refinamentos naturalmente. Duas de suas características principais são a existência de um mecanismo capaz de representar unicamente cada simplexo da triangulação e a existência de regras algébricas para percorrer a estrutura. O uso dessas regras previne a estrutura de necessitar armazenar informações de conectividade dos simplexos, logo, capacitando um armazenamento mais eficiente.

A triangulação J_1^a consiste em uma grade computacional formada por hipercubos (*blocos*) n-dimensionais no domínio R^n. Cada bloco é dividido por $2^n n!$ n-simplexos que

podem ser descritos algebricamente usando a sêxtupla:

$$S = (g, r, \pi, s, t, h).$$

Os dois primeiros elementos de S definem em que bloco o simplexo está contido, sendo g um vetor n-dimensional que indica as coordenadas do bloco em um nível de refinamento particular r da grade. A Figura 2.8 ilustra, à esquerda, uma grade bidimensional da J_1^a e, à direita, um bloco destacado desta grade com nível de refinamento $r = 0$ (0-bloco) e $g = (3,2)$. Também na figura 2.8, pode ser notado que os blocos da grade mais escuros são de nível $r = 1$ (assim chamado de 1-bloco) e, por isso, fazem parte de uma região da grade de maior resolução.

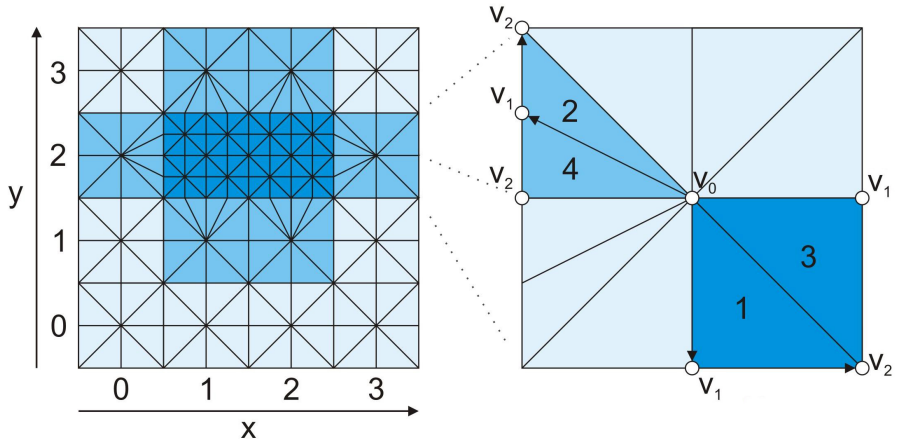


Figura 2.8: Exemplo 2D de uma grade da triangulação J_1^a (esquerda) e detalhes do bloco $g = (3,2)$, $r = 0$ onde é mostrado dois percursos para traçar os simplexos do bloco (direita).

A triangulação J_1^a foi utilizada com muita eficiência em aplicações de geração de malhas triangulares [Gois et al. 2007] e geração de malhas tetraédricas [da Cunha et al. 2008]. A Figura 2.9 ilustra uma malha gerada através da triangulação J_1^a . Porém no escopo do nosso trabalho utilizaremos a triangulação J_1^a somente quanto a sua estrutura de grade básica e suas regras de transição entre essas estruturas. Tendo isso em vista, chamaremos essa estrutura de **Estrutura** J_1^a .

Para maiores detalhes das regras algébricas pertencentes a triangulação J_1^a consulte o trabalho [Castelo et al. 2006].

2.4 Redução e Codificação de Dados Gráficos

Malhas 3D têm sido amplamente utilizadas em aplicações gráficas para representação de objetos 3D. Eles frequentemente requerem uma grande quantidade de dados para armazenamento e/ou transmissão no formato bruto dos dados. Como a maioria das aplicações exigem armazenamento compacto, transmissão rápida e processamento eficiente

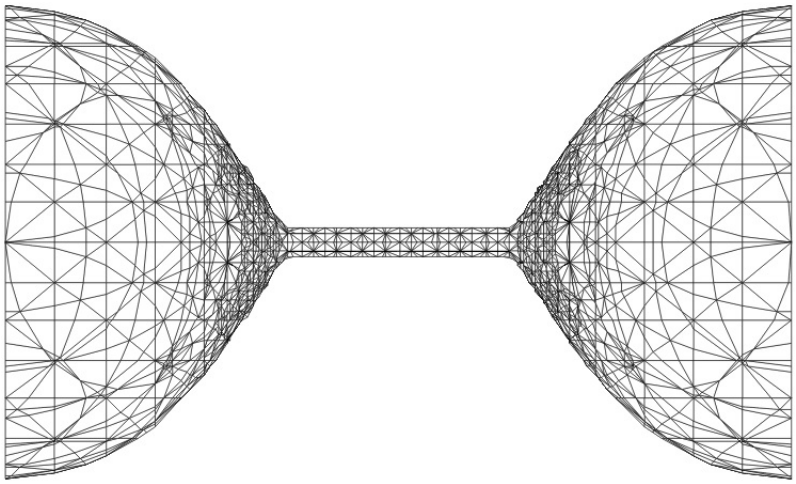


Figura 2.9: Exemplo de uma triangulação 3D obtida usando a triangulação J_1^a . Figura obtida do trabalho [Castelo et al. 2006].

de malhas 3D, foram propostos vários algoritmos para comprimir malhas 3D de forma eficiente desde início da década de 90 [Peng et al. 2005].

Um típico algoritmo de compressão de malha codifica dados de conectividade e geométricos separadamente. Inicialmente, os trabalhos se centraram em codificação de conectividade. Em seguida, a codificação de dados geométricos foi determinada pela codificação de conectividade subjacente. No entanto, já que dados geométricos exigem mais bits do que os dados de topologia, vários métodos têm sido propostos recentemente para compressão eficiente de dados de geometria sem referência aos dados de topologia.

Infelizmente, os métodos de codificação geométrica resultam em malhas diferentes das malhas iniciais. E como não temos interesse em modificar a malha inicial, focaremos somente em métodos de codificação por conectividade. São seis os tipos de codificação por conectividade: conjunto de faces indexadas, *triangle strips* (tira triangular), árvore de expansão (*spanning tree*), decomposição por camadas, abordagem orientada por valência e conquista triangular. A seguir descrevemos cada tipo.

2.4.1 Conjunto de faces indexadas

Uma malha triangular é representada por um conjunto de faces indexadas que consiste em uma lista de coordenadas e uma lista de faces. A lista de coordenadas apresenta as coordenadas de todos os vértices, e lista de faces apresenta cada face ao indexar seus vértices contidos na lista de coordenadas. Por exemplo, a Figura 2.10.a ilustra uma malha e sua lista de faces.

2.4.2 Tiras triangulares - *Triangle strip*

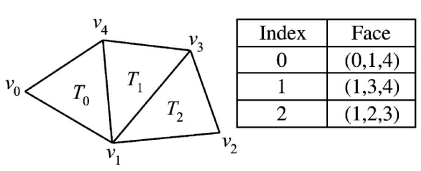
O método de *triangle strip* tenta dividir uma malha 3D em longas tiras de triângulos, e depois codificar estas tiras [Taubin & Rossignac 1998]. O principal objetivo desse

método é reduzir a quantidade de dados transmitidos entre a CPU e a placa gráfica, uma vez que as tiras triângulo são bem suportados pela maioria das placas gráficas. Embora este esquema exige menos espaço de armazenamento e de banda de transmissão do que a representação por conjuntos de faces indexadas, este método ainda não é muito eficiente para o propósito de compressão. Foram propostas algumas variações desse método em [Deering 1995] e[Taubin & Rossignac 1998]; Chow [Chow 1997] propôs uma abordagem compressão a partir de tiras para optimizar *rendering* em tempo real.

A Figura 2.10.b ilustra três tipos de tiras.

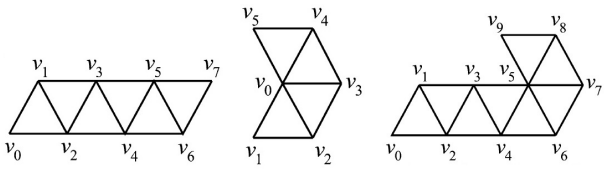
2.4.3 Árvore de expansão - *Spanning tree*

Em Turán [Turán 1984] foi observado que a conectividade de um grafo planar pode ser codificado com um número constante de bits por vértice usando duas árvores de expansão: uma árvore de expansão para vértices outra para triângulos (ilustrado na Figura 2.10.c). Com base nesta observação, Taubin e Rossignac [Taubin & Rossignac 1998] apresentaram uma abordagem topológica cirúrgica para codificar a conectividade da malha. A idéia básica é cortar uma determinada malha ao longo de um conjunto de arestas de corte selecionadas a fim de gerar um polígono plano. A conectividade da malha é, então, representada pelas estruturas das arestas de corte e dos polígonos. Em uma malha simples, qualquer árvore de expansão de vértice pode ser selecionada como o conjunto de arestas de corte.

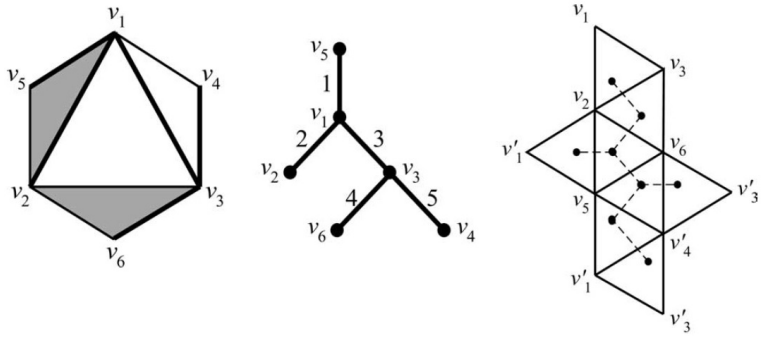


Index	Face
0	(0,1,4)
1	(1,3,4)
2	(1,2,3)

(a) Conjunto de faces



(b) *Strip* triangular



(c) *Spanning tree*

Figura 2.10: Ilustração de três tipos de codificação de malhas: (a) conjunto de faces indexadas, (b) *triangle strips* e (c) *spanning tree*.

2.4.4 Decomposição por camadas - *Layered decomposition*

Em [Bajaj et al. 1999] foi apresentado um método de codificação de conectividade usando uma estrutura de vértices em camadas. Esta estrutura decompõe uma malha triangular em várias camadas concêntricas de vértices, e, em seguida, constrói camadas triangulares dentro de cada par de camadas de vértices adjacentes. A conectividade da malha é representada pelo número total de camadas de vértice, o layout de cada camada vértice, e o layout dos triângulos em cada camada de triângulos. Idealmente, uma camada de vértice não se intercepta e uma camada de triângulo é uma tira de triângulos generalizada. Em tal caso, a compressão de conectividade é reduzida à codificação do número de camadas de vértice, o número de vértices de cada camada de vértice, e a tira de triângulo generalizada em cada camada de triângulo. No entanto, na prática, artefatos são introduzidos devido à existência de pontos de ramificação, "triângulos bolhas", e leque de triângulos. Em [Taubin & Rossignac 1998], este método foi utilizado para a construção de árvore de expansão de vértices.

No seu trabalho, Bajaj et al. [Bajaj et al. 1998] propuseram um método de decomposição por camadas codificar modelos CAD grandes, além disso, é também capaz de codificar malhas quadriláteras e poligonais gerais.

2.4.5 Abordagem orientada por valência - *Valence-driven approach*

Inicialmente proposta por Touma e Gotsman [Touma & Gotsman 1998], a abordagem orientada a valência começa a partir de um triângulo semente cujas três arestas formam a fronteira inicial. Essa fronteira divide toda a malha em duas partes, isto é, a parte interna que foi processada e a parte externa, que está para ser processada. Em seguida, a fronteira gradualmente se expande para fora, até que toda a malha ser processada. O resultado desse método uma sequência de valências de vértices, a partir da qual a conectividade original pode ser reconstruída. Posteriormente, [Alliez & Desbrun 2001] propuseram uma outra abordagem desse método que melhora sua performance.

2.4.6 Conquista triangular - *Triangle conquest*

Gumhold e Straßer [Gumhold & Strasser 1998] foram os primeiros a propor uma abordagem de conquista triangular, chamada de máquina de corte de fronteira (*cut-border machine*). Semelhante à abordagem orientada a valência, a abordagem conquista triangular começa a partir da fronteira inicial, que divide toda a malha em partes conquistadas e não conquistadas, e insere triângulo por triângulo na região conquistada. A principal diferença é que a abordagem conquista triangular tem como saída a operação de construção de novos triângulos, enquanto que a abordagem orientada a valência produz as valências de novos vértices. Em [Gumhold 1999], Gumhold melhorou a performance de compressão do algoritmo ao propor um codificador aritmético e otimizando a codificação de fronteira.

Rossignac [Rossignac 1999] propôs o algoritmo *edgebreaker*, que é um outro exemplo da abordagem conquista triangular. É quase equivalente à máquina de corte de fronteira, exceto que ele não codifica os dados de deslocamento associados à operação de divisão.

Em [King & Rossignac 1999] o algoritmo *edgebreaker* foi modificado para garantir boa performance no pior caso.

A codificação do conjunto de primitivas visíveis através do uso de *triangle strips* ou *spanning tree* pode auxiliar em uma redução ainda maior da quantidade dados sendo processados durante a visualização de uma cena, especialmente do ponto de vista da transmissão de dados.

2.5 Estruturas de Dados para Visualização

No foco do nosso trabalho, a representação de um modelo é a discretização de sua geometria e atributos através de malhas. Estruturas de dados (ED) topológicas buscam indexar as primitivas que compõem uma dada malha de modo a representar as relações de incidência e adjacência entre tais primitivas, garantindo ainda um acesso eficiente a suas informações [Mäntylä 1988] [Bottasso et al. 1998]. Muitas das estruturas topológicas descritas na literatura visam indexar os dados de modo a facilitar os mecanismos de acesso as relações de vizinhança e as buscas empregadas na construção ou leitura das malhas [Kwok et al. 1995], viabilizando ainda o controle de procedimentos adaptativos.

Para auxiliar o visualizador, a estrutura de dados topológica *Mate Face* (MF) (proposta por [Ícaro L. L. da Cunha 2009]) será utilizada como estrutura de representação e armazenamento de malhas carregadas no visualizador deste trabalho.

2.6 Imageamento Médico por Ressonância Magnética

Radiologistas usam a ressonância magnética para medir o volume de estruturas no cérebro. Quando uma pessoa tem a doença de Alzheimer, a ressonância magnética pode mostrar uma diminuição no volume global do hipocampo (uma área do cérebro que desempenha um papel crítico na memória). Mas a ressonância magnética não é capaz de mostrar mudanças no cérebro que se desenvolvem antes do hipocampo começar a encolher. É aí que o imageamento por difusão curtótica (*Diffusion Kurtosis Imaging* - DKI) entra em cena.

Em um Imageamento por Difusão Tensorial (Diffusion Tensor Imaging - DTI) convencional, a distribuição de difusão da água é descrita como um tensor de difusidade tridimensional de segunda ordem. Assume-se que a difusão ocorre em um ambiente livre e sem restrições, com uma distribuição gaussiana do deslocamento de difusão e, consequentemente, o sinal de difusão ponderada (diffusion weighted - DW) decai com fator de difusão (valor/peso b) monoexponencialmente. No tecido biológico, microestruturas celulares complexas fazem a difusão de água um processo altamente impedido ou limitado. Decaimentos não-monoexponenciais são observados experimentalmente tanto em massa branca e cinzenta. Como resultado, quantificação por DTI é dependente no valor-b e DTI falha para utilizar plenamente as medidas de difusão que são inerentes à microestrutura do tecido.

O tensor de difusão foi originalmente proposto para uso em imagens de ressonância magnética (MRI) por Peter Basser em 1994 [Basser et al. 1994*a*] [Basser et al. 1994*b*].

Antes de DTI, difusão em MRI [Le Bihan et al. 1986] [Le Bihan & Basser 1995] tinha sido desenvolvida a partir de pesquisa em difusão por ressonância magnética nuclear [Le Bihan 1991]. Antes da introdução do modelo de tensor de difusão, para medir a difusão anisotrópica a orientação dos axônios em uma amostra de tecido tinha de ser conhecida, de modo que apenas as amostras fixas, tais como o axônio da lula gigante poderiam ser digitalizados [Beaulieu & Allen 1994]. A introdução do modelo de tensor de difusão permitiu, pela primeira vez, uma descrição invariante da forma de difusão de água. A invariância a rotação foi fundamental porque permitiu a aplicação do método DTI à anatomia complexa dos feixes de fibras no cérebro humano [Pierpaoli et al. 1996]. No entanto, deve-se notar que o tensor de difusão não é capaz de descrever completamente passagem dos feixes de fibras [Wiegell & Wedeen Van 2000] [Tuch 2002].

A popularidade do DTI tem sido enorme. Foi aplicada a uma enorme variedade de estudos neurocientíficos (veja as revisões em [Horsfield & Jones 2002] [Ciccarelli et al. 2008] [Assaf & Pasternak 2008]), incluindo esquizofrenia [Kubicki et al. 2007], lesão cerebral traumática [Maller et al. 2010], esclerose múltipla [Inglese & Bester 2010], autismo [Lange et al. 2010], e envelhecimento [Westlye et al. 2010].

O tensor de difusão (DT) descreve a difusão de moléculas de água utilizando um modelo gaussiano. Tecnicamente, é proporcional à matriz de covariância de uma distribuição gaussiana tridimensional que modela os deslocamentos das moléculas. O DT é uma matriz 3×3 simétrica definida positiva, e essas propriedades da matriz significa que ele tem 3 autovetores ortogonais (perpendiculares entre si) e três autovalores positivos. O autovetor principal do tensor de difusão aponta na direção principal de difusão (a direção da difusão mais rápida). Em tecidos fibrosos anisotrópicos o principal autovetor também define o eixo de traçado da fibra do tecido [Basser et al. 1994*b*], e, assim, os três autovetores ortogonais podem ser pensados como um sistema de coordenadas de uma fibra local. (Note que esta interpretação é apenas estritamente verdadeira em regiões onde feixes de fibras não se cruzam, formam um leque, ou ramificam.) Os três autovalores positivos do tensor $(\lambda_1, \lambda_2, \lambda_3)$ dão a difusividade na direção de cada autovetor. Juntos, os autovetores e autovalores definem um elipsoide que representa um isosurface de probabilidade de difusão (gaussiana).

Para medir a difusão pela ressonância magnética, gradientes de campo magnético são empregados para criar uma imagem que está sensibilizada para a difusão em uma direção particular. Repetindo este processo de ponderação de difusão em múltiplas direcções, um modelo de difusão tridimensional (tensor) pode ser estimado. Em termos simplificados, imagem de difusão funciona através da introdução de pulsos extras de gradiente cujo efeito "se cancela" para as moléculas de água estacionárias, e provoca uma mudança de fase aleatória para moléculas que se difundem. Devido à sua fase aleatória, sinal de difusão de moléculas é perdido. Esta perda de sinal cria voxels mais escuros (pixels volumétricos). Isso significa que feixes de fibras de massa branca paralelas à direção do gradiente aparecerá escura na imagem de difusão ponderada para essa direção.

As Equações para realizar o cálculo do tensor de difusão estão presentes em [Assaf & Pasternak 2008].

DTI é geralmente exibido ou por condessando as informações contidas no tensor em um número (escalar) ou em 4 números (para gerar uma cor em RGB cor e um valor de

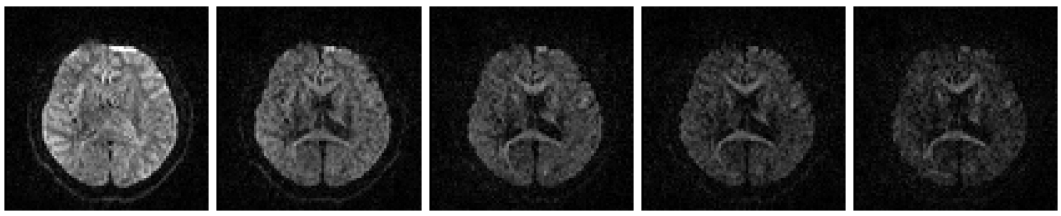
brilho. O tensor de difusão também pode ser visualizado usando glifos, que são pequenas representações 3D do maior autovetor ou todo tensor. Finalmente, DTI é muitas vezes visto através da estimativa do curso de traçados da fibra da massa branca através do cérebro por meio de um processo chamado tractografia.

2.6.1 Imageamento por Difusão Curtótica (*Diffusion Kurtosis Imaging* - DKI)

DKI fornece informação quantificável no comportamento não-Gaussiana da difusão da água difusão no tecido biológico. Tendo isso em vista, propomos desenvolver uma ferramenta de visualização que seja capaz de mostra o comportamento da estrutura cerebral dadas as informações obtidas pelo DKI. Atualmente essas informações são visualizadas a partir de diversas imagens de uma dada fatia do cérebro, queremos integrar toda a informação de tal forma que sejamos capaz de visualizá-la em 3D e de modo interativo (alterando os parâmetros de dados desejáveis).

DKI caracteriza difusão restrita e pode ser facilmente implementado na maioria dos *scanners* clínicos. Ele fornece uma descrição de alta ordem do processo de difusão de água através de um tensor de difusidade 3D de segunda ordem como em DTI convencional, juntamente com um tensor de curtose 3D de quarta-ordem. Devido ao fato de curtose ser uma medida do desvio do perfil de deslocamento da difusão a partir de uma distribuição de gaussiana, análise de dados DKI quantifica o grau de restrição de difusão ou a complexidade do tecido sem qualquer suposição biofísica.

Imagens (exemplificadas na Figura 2.11 e cujos valores representam o sinal de difusão ponderado) foram obtidas para todos os cinco valore de intensidade b (b=0, 500, 1000, 1500, 2000, 2500 s/mm^2) e 30 direções. A Figura 2.12 ilustra a intensidade de sinal de cada direção de cada voxel (de uma determinada fatia) para b=1500 s/mm^2.



(a) b=500 (b) b=1000 (c) b=1500 (d) b=2000 (e) b=2500

Figura 2.11: Ilustração de uma das fatias do cérebro para um dado vetor de obtenção de imagem nas 5 intensidades b diferentes. A partir dessa ilustração, pode-se notar que intensidade b influencia a intesidade do sinal obtida durante o imageamento.

A partir da equação de mínimos quadrados não-lineares abaixo, poderemos determinar os parâmetros de difusão e curtose para esse conjunto de imagens.

$$lnS(\mathbf{g},b) = lnS_0 - b\sum_{i=1}^{3}\sum_{j=1}^{3} g_i g_j D_{ij} + \frac{1}{6}b^2\sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{3}\sum_{l=1}^{3} g_i g_j g_k g_l K_{ijkl}$$

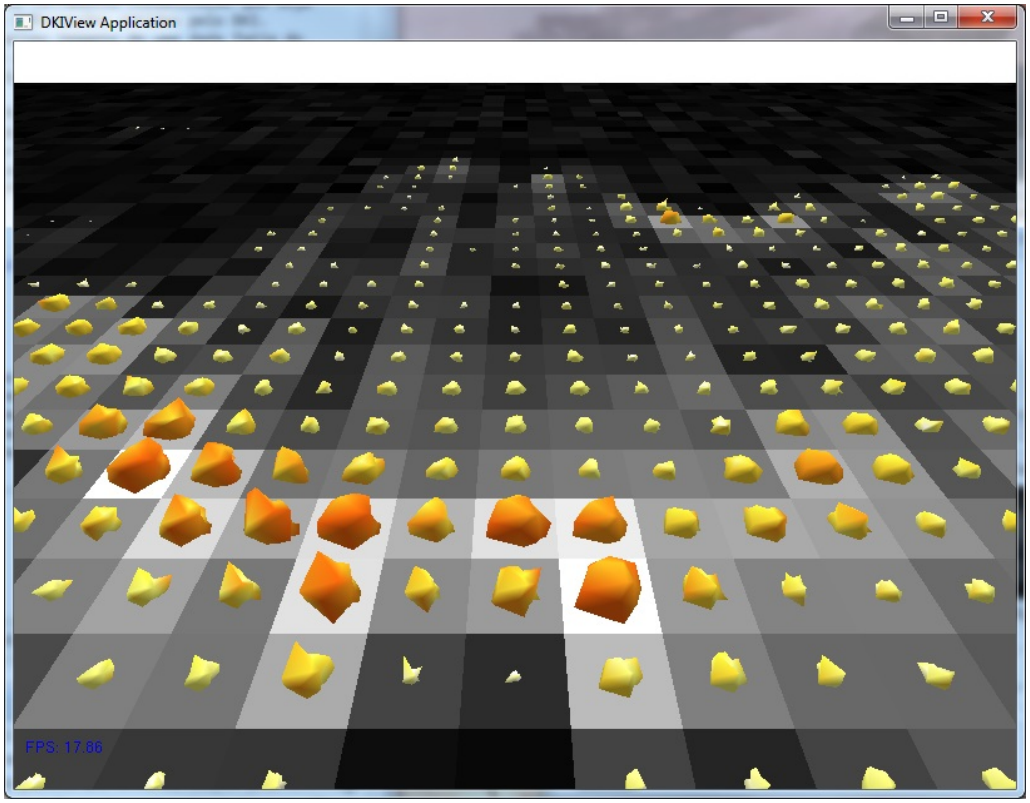


Figura 2.12: Ilustração das esferas de difusão para cada ponto da imagem com intensidade b igual a 1500 s/mm^2. Essas esferas ilustram a proporção de difusão das 30 direções de captura para cada voxel da fatia visualizada.

onde,

- $\mathbf{g} = (g_1, g_2, g_3)$ é o vetor direcional do gradiente de difusão;
- $S(\mathbf{g}, b)$ é o sinal ponderado por difusão a partir de um valor b e direção $\mathbf{g}$. Em outras palavras, é o valor de cada voxel da imagem volumétrica.
- S_0 é o sinal de ressonância magnética sem ponderação de difusão ($b = 0s/mm^2$) e é a média de todos os cinco volumes $b = 0$ que foram adquiridos.
- D_{ij} é elemento do tensor 3x3 de difusão $\mathbf{D}$.
- K_{ijkl} é o elemento do tensor 3x3x3x3 de quarta ordem. K_{ijkl} está relacionado com os elementos W_{ijkl} do tensor de difusão catódica $\mathbf{W}$ e a difusidade média MD (mm^2/s) pela equação:

$$K_{ijkl} = MD^2.W_{ijkl}$$

As matrizes $\mathbf{D}$ e $\mathbf{K}$ são totalmente simétricas com 6 valores independentes do tensor de difusão e 15 do tensor de curtose (KT). Se faz necessário calcular difusidade aparente ($D_{app}(\mathbf{g})$) e a curtose aparente ($K_{app}(\mathbf{g})$). Os três autovalores $\lambda_1, \lambda_2, \lambda_3$ ($\lambda_1 \geq \lambda_2 \geq\geq \lambda_3$)

e o correspondente autovetor (e_1, e_2, e_3) foram derivados por decomposição de DT. Uma série de parâmetros de difusividade podem ser calculados a partir desses valores, tais como: difusidade média, anisotropia fracionada, difusividade axial e difusividade radial.

Os parâmetros relacionados à difusão curtótica são derivados a partir de KT. A curtose média (MK) é calculada tirando a média de K_{app} em todas as N=30 direções:

$$MK = \frac{1}{N}\sum_{i=1}^{N}(K_{app})_i$$

A Figura 2.13 ilustra uma comparação da distribuição de curtose com a distribuição de difusão. Se a difusão é modelada como um elipsóide com sua direção principal (definida por v_1) apontando ao longo dos axons da massa branca, então a distribuição de curtose será como uma panqueca em forma de um elipsoide. A curtose é achatada na direção axial do elipsoide difusão (definida por v_1), porque a difusão é mais livre ao longo dos axons, resultando numa distribuição de deslocamento de difusão mais gaussiano. A curtose é alta na direção radial do elipsoide difusão (o plano gerado por v_2 e v_3), porque o movimento das moléculas de água é altamente heterogênea, devido às bainhas de mielina e membranas celulares, etc., resultando numa distribuição de deslocamento altamente não-gaussiana. Note, porém, que a representação da distribuição de curtose por um elipsoide é uma visão simplificada de uma estrutura mais complexa definida pelo tensor de curtose de quarta ordem. A teoria por trás de DKI é mais detalhada em [Jensen et al. 2005] e [Zhuo et al. 2012].

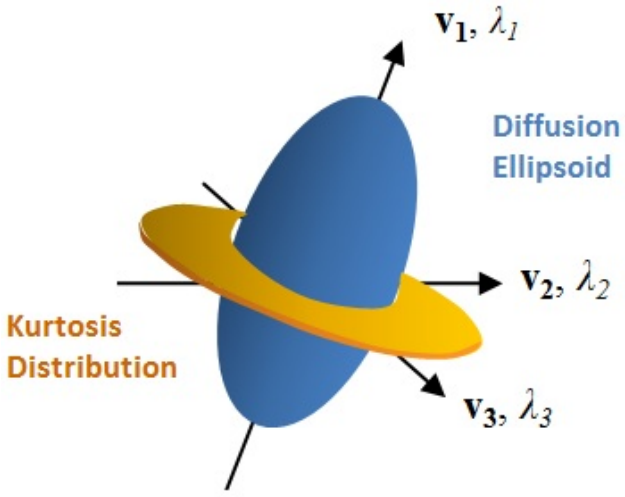


Figura 2.13: Ilustração da distribuição de difusão e de curtose em um sistema 3D definido pelo autovetores de difusão (v_1, v_2, v_3). A distribuição de difusão é o elipsoide (azul) com a direção principal apontando para v_1. A distribuição de curtose está em forma achatada (amarelo) com a curtose maior ao longo da direção radial do elipsoide de difusão.

Em [Jensen et al. 2005], é demostrado que DKI oferece uma abordagem mais compreensiva que DTI na descrição do complexo processo de difusão de água dentro de um

organismo vivo. Ao estimar ambas difusidade e curtose, pode fornecer melhor sensibilidade e especificidade na caracterização da difusão (em ressonância magnética) de tecidos neurais. E, por isso, é capaz de detalhar não só massa branca mas também a cinzenta.

Embora DKI é um método de imageamento relativamente novo e houve apenas um número limitado de estudos que utilizam DKI, parâmetros de curtose tem mostrado ter, potencialmente, maior sensibilidade na caracterização de tecidos neurais que DTI. Cheung et al. descobriu que os parâmetros DKI foram mais sensíveis às mudanças de desenvolvidas em ratos do que os parâmetros convencionais DTI [Cheung et al. 2009]. Falangola et al. relataram mudanças na MK relacionadas com a idade, que nunca antes foram relatadas em estudos de envelhecimento em DTI [Falangola et al. 2008].

Capítulo 3

Trabalhos Relacionados

Como o *culling* por *view frustum* e por *back-face* são triviais para se determinar, a literatura recente, até o momento, tem sido quase sempre focada em culling por oclusão. Um dos mais importantes trabalhos que encontramos nesta área é o *survey* realizado por [Cohen-Or et al. 2003]. Eles subdividiram e classificaram as várias técnicas de *culling* por oclusão desenvolvidas, proporam uma classificação dos vários métodos, de acordo com as suas características, como pode ser visto na Seção 3.1.

3.1 Classificação

Os métodos de determinação de oclusão podem ser classificados em *point based* (baseado em pontos) ou *region based* (baseado em regiões). Os métodos baseados em ponto executam seus cálculos a partir da perspectiva do ponto. Do outro lado, métodos baseados em região efetuam suas computações a partir de um ponto de vista global que é válido de qualquer região da cena [Cohen-Or et al. 2003].

Os métodos baseados em ponto são classificados como:

- **Métodos de Precisão de Imagem** - operam com uma representação discreta dos objetos quando estes são quebrados em fragmentos durante o processo de rasterização. Entre esse tipo os mais comuns são os baseados em Traçado de Raios [Bala et al. 1999*b*], [Bala et al. 1999*a*], [Cohen-or & Shaked 1995], [Cohen-Or et al. 1996], [Parker et al. 1999]; e os baseados em um *Z-buffer* hierárquico [Greene et al. 1993], [Greene & Kass 1994], [Meagher 1982], [Greene 1999], [Greene 2001*a*], [Greene 2001*b*].
  Em [Zhang 1998], um método de mapeamento hierárquico de oclusão simular aos métodos anteriores foi apresentado, calcula-se a visibilidade testando sobreposição e profundidade. Outro método, proposto por [Bernardini et al. 2000], gera oclusores eficientes para um dado ponto de vista e recursivamente removem nós de octree oclusos. Bartz et al. [Bartz et al. 1999] apresenta uma técnica de representação hierárquica da cena que é testada contra partes oclusas da imagem. Klosowski e Silva propuseram dois métodos ([Klosowski & Silva 2000] [Klosowski & Silva 2001]) que testam visibilidade volumétrica o mais recente deles estende a abordagem numa técnica conservativa ao utilizar técnicas baseadas em precisão de imagem. Finalmente, em [Wonka & Schmalstieg 1999] foi proposto método que generaliza cenas

3D em cenas de 2.5D, com isso calculam a oclusão em respeito a um ponto de vista usando *Z-buffer*.

- **Precisão do objeto** - utiliza os puros objetos para realizar cálculo de visibilidade. Entre esse métodos citamos os trabalho de [Luebke & Georges 1995] e [Jones 1971] que utilizam portais para calcular visibilidade. Coorg e Teller [Coorg & Teller 1997] computam a oclusão causada por um grande subconjunto de oclusores convexos. Em [Hudson et al. 1997], foi proposta uma abordagem baseada em escolher dinamicamente o conjunto de oclusores. E Bittner et al. [Bittner et al. 1998] estenderam essa abordagem usando árvores BSP.

E, os métodos baseados em região podem ser classificado como:

- ***Cell-and-Portal*** **(célula e portal)** - Inicializa com um conjunto vazio de primitivas visíveis e o preenche a partir células visíveis através de portais. Desses métodos, destacam-se os trabalhos propostos por [Airey et al. 1990] e [Teller & Séquin 1991].
- **Região Genérica** - Inicialmente assume que todas as primitivas são visíveis e posteriormente elimina uma a uma a medida que se descobre que estão escondidas. Cohen-Or et al. [Cohen-Or et al. 1998] propuseram um método conservativo baseado na oclusão causada por objetos individuais convexos e grandes. Schaufler et al. [Schaufler et al. 2000] introduz outro método conservativo para computação de visibilidade de células de visualização. Em [Durand et al. 2000], é apresentada uma extensão aos métodos de precisão de imagem baseado em ponto, como o *Z-buffer* hierárquico, para calcular a visibilidade de uma célula no preprocessamento de computação de primitivas pontencialmente visíveis (*Potencially Visible Set*- PVS). Koltun et al. [Koltun et al. 2000] introduziram a noção de oclusores virtuais baseados em região e propuseram uma implementação 2.5D e em [Koltun et al. 2001] introduzem um método que melhora o desempenho de técnicas baseadas em região quanto à precisão e rapidez. Finalmente, Wonka et al. apresentaram [Wonka et al. 2000] uma abordagem baseada na possibilidade de se calcular uma aproximação conservativa de "células sombra" a partir de um conjunto discreto de pontos de amostragem localizados na fronteira de células de visualização.

Em [Cohen-Or et al. 2003] outros critérios de classificação foram propostos, tais como:

1. Se o método está restrito a uma planta baixa 2D da cena ou se é capaz de lidar com toda a cena 3D para determinação de visibilidade;
2. Se um método superestima o conjunto visível (abordagem conservativa) ou se esse método aproxima esse conjunto. No caso dos métodos conservativos, deseja-se sempre verificar qual o grau de superestimação. Nem sempre a superestimação é um fator ruim, só depende de o quanto o seu grau é elevado.;
3. Se o método trata oclusão causada por todos os objetos da cena ou se só utiliza um conjunto selecionado de oclusores. Pode também ser utilizados portais, subconjunto de objetos e conjunto de convexos;

4. Se cada primitiva oclusora é tratada como um grupo de oclusores para ser mais preciso ou individualmente;
5. Se o método executa uma etapa de precomputação antes de inicializar a visualização. Algumas operações podem ser realizadas como seleção de oclusores, uso de volumes, uso do PVS, banco de dados de oclusores, etc. Geralmente, esta etapa é importante pois só é executada uma vez, evitando assim processamento de operações desnecessárias durante a etapa de visualização;
6. Se é necessário o uso de um hardware especial para melhorar o desempenho da etapa de precomputação ou até mesmo a etapa de renderização (uso de *Z-buffer* ou buffer secundário);
7. E se é capaz de tratar cenas dinâmicas, onde um ou mais objetos se deslocam ao longo da cena. Nesse caso, operações como atualização de hierarquia, uso volume de deslocamento de oclusores ou ignoração de oclusão causada pelo objeto em movimento podem ser utilizados.

A tabela 3.1 apresenta um resumo comparativo dos métodos apresentados.

3.2 Abordagens Recentes

Bittner et al. [Bittner et al. 2009] propuseram um método baseado em região de visibilidade, onde raios são lançados para gerar uma amostragem de visibilidade e utiliza a informação de cada raio para todas as células de visualização que intersecta. Este método usa uma estratégia de amostragem adaptativa baseada na mutação de raios que exploram a coerência da visibilidade. Chandak et al. [Chandak et al. 2009] propuseram um método relativamente semelhante ao nosso, a abordagem deles utiliza uma quantidade alta de blocos (em forma de frustum) e computa oclusão usando simples testes de interseção. Eles usam este método para calcular com precisão os caminhos de reflexão a partir de uma fonte de som do ponto. O método proposto por Tian et al. [Tian et al. 2010] integra simplificação baseada em amostragem adaptativa, a determinação de visibilidade, gerenciamento de dados *out-of-core* e *level-of-detail*-LOD (nível de detalhes). Durante o preprocessamento, os objetos são subdivididos e uma hierarquia de agrupamento de volumes delimitadores é construída. Eles fazem uso do voxels adaptativos, que um novo método de amostragem adaptativo aplicado para gerar modelos de LOD. Antani et. Al. [Antani et al. 2010] introduziram um rápido algoritmo de seleção de oclusor que combina pequenos triângulos adjacentes para formar grandes oclusores e realizar cálculos conservadores na precisão objeto-espaço. Este método é aplicado para cálculo de propagação do som, melhorando o desempenho de algoritmos de difração de arestas por um fator de 2 a 4.

3.3 Contextualização do Trabalho

Nosso método pode ser classificado como um método baseado em região, e é semelhante ao tipo *cell-and-portal*. A única diferença é que, ao invés de usar um conjunto

de células, usamos uma grade de subdivisão de cena; e ao invés de usar um portal, usamos a adjacência entre blocos da grade. Experimentalmente, essa abordagem se mostrou também, no mínimo, útil.

Seguindo os outros critérios citados em [Cohen-Or et al. 2003], o método que propomos:

- Superestima o conjunto de primitivas potencialmente visíveis (discutimos sobre esse fato no Capítulo 4 e analisamos seu grau de superestimação na Seção 6.3);
- Mesmo que o nosso método seja baseado em blocos, ele trata também oclusão criada entre blocos adjacentes. Em outras palavras o tratamento de oclusão pode ser global além de ser local (oclusão ocorrida em cada bloco);
- Lida com cenas 3D e, devido ao uso da estrutura base da J_1^a . Além disso, acreditamos que poderemos adicionar uma quarta dimensão para aplicações com objetos e cenas dinâmicos;
- A etapa de precomputação é executada sem a necessidade de um hardware especial. Mesmo assim, o uso de um processador gráfico dedicado melhoraria o desempenho dessa etapa, porém isso não é crucial já que só executada uma única vez;
- Até o momento, lidamos com cenas dinâmicas ao ignorar a oclusão causada pelos objetos em movimento. Devido a isso, não mostraremos resultados baseado em objetos dinâmicos. Acreditamos que seja possível estender nossa estrutura de tal forma que possamos detectar oclusões locais (oclusão interna dos blocos onde os objetos se encontram), bastaria adicionar uma quarda dimensão à grade da estrutura.

No Capítulo 6, apresentamos uma análise comparativa entre o nosso método e alguns dos métodos apresentados neste capítulo. A escolha de quais métodos que iremos comparar deve ser de acordo com as suas características e que sejam semalhantes ao nosso método. Por exemplo, não queremos comparar nosso método com aqueles que estão restritos a uma planta baixa 2D da cena porque não seria uma comparação justa no quesito de superestimação. Outro exemplo também é a comparação com métodos baseados em portais pois estes métodos só são eficientes em cenas de ambientes internos.

Tabela 3.1: Comparação entre os métodos apresentados e o nosso método proposto a partir classificação apresentada na seção 3.1.

Método	2D/3D	Conserv.	Oclusor	Fusão	PreComp	Hardware	Dinam.
Cell-and-portal							
Airey	2D/3D	sim/não	portais	-	PVS	não	não
Teller	2D/3D	sim	portais	-	PVS	não	não
Precisão do objeto							
Luebke & George	3D	sim	portais	-	sim	não	hierarquia
Coorg & Teller	3D	sim	convexo	silhueta	seleção	não	hierarquia
Hudson et al.	3D	sim	convexo	não	seleção	não	hierarquia
Bittner et al.	3D	sim	grande	sim	seleção	não	hierarquia
Precisão de imagem							
Greene et al.	3D	sim	todos	sim	não	sim	sim
Zhang et al.	3D	sim/não	subconj.	sim	banco	hardware	hierarquia
Bernardini et al.	3D	sim	preproc.	sim	oclusor	não	não
Bartz et al.	3D	não	todos	sim	não	buffer	hierarquia
Klosowski & Silva 00	3D	não	todos	sim	volume	não	hierarquia
Klosowski & Silva 01	3D	não	todos	sim	volume	sim	hierarquia
Wonka et al. 99	2.5D	sim	subconj.	sim	não	*Z-buffer*	sim
Região genérica							
Cohen-Or et al.	3D	sim	todos	não	PVS	não	volume
Schaufler et al.	2D/3D	sim	todos	sim	PVS	não	volume
Durand et al.	3D	sim	subconj.	sim	PVS	sim	volume
Koltun et al. 00	2D/2.5D	sim	subconj.	sim	oclusor	não	volume
Wonka et al. 00	2D	sim	subconj.	sim	PVS	sim	volume
Koltun et al. 01	2.5D	sim	todos	sim	não	sim	hierarquia
Nosso método proposto							
ED Vis.	3D	sim	todos	sim	grade	não	ignora

Capítulo 4

O Problema

Apresentamos a formulação matemática do problema de visibilidade tridimensional e propomos uma solução deste problema. Além disso, apresentamos também uma solução para o problema de geração de malhas da estrutura cerebral e sua visualização.

4.1 Formulação Matemática

A necessidade de se gerar cenas cada vez mais complexas para melhorar o realismo cria um problema quanto ao desperdício de processamento. Isso se deve ao fato que, mesmo que uma maior quantidade de primitivas possibilite uma representação mais fiel, sempre haverá primitivas que não estão sendo visualizados a partir de certos pontos de vista da cena. Processar primitivas não visíveis causa um desperdício de processamento, de armazenamento e (em aplicações web) de banda de rede.

O problema que então queremos resolver é reduzir a quantidade de primitivas carregadas para visualização sem modificar a forma original dos objetos gráficos. Então o problema se resume a:

Seja T o conjunto (de tamanho n) de todas as primitivas que compõem a cena a ser visualizada, para toda primitiva $t_i \in T$ $(0 \leq i < n)$ deve-se verificar se:

1. t_i está contido no volume de visualização;
2. Se a normal de t_i está voltada para a posição de visualização e;
3. Se t_i não é escondido desse ponto de visualização, em outras palavras, se não existe t_j ou um subconjunto de T que não são oclusores de t_i.

Se t_i passar por todas essas condições então, t_i pode ser carregado para visualização.

4.2 Modelagem

Como foi dito anteriormente, para reduzir processamento desnecessário, desejamos eliminar o máximo possível os dados (primitivas) que não contribuem para a visualização da cena. Ou seja, como ilustrado na Figura 4.1, desejamos somente processar as arestas contínuas.

A seguir, vamos discutir duas operações onde a redução de dados gráficos se faz necessária.

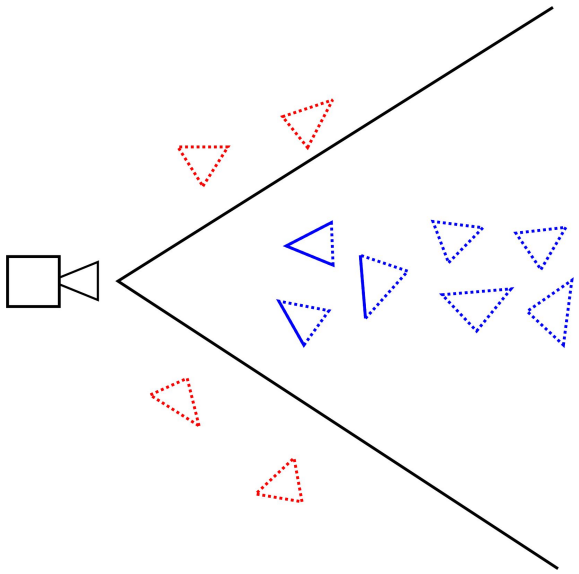


Figura 4.1: Ilustração de uma cena onde a partir do ponto de vista da câmera certas primitivas são visíveis (arestas contínuas) e outras não (arestas tracejadas). Desejamos encontrar dentre todas as primitivas que compõem a cena quais primitivas são visíveis.

4.2.1 Renderização de dados gráficos

Falamos na Seção 2.2 que uma quantidade relativamente grande de dados pode afetar o desempenho das operações de visibilidade executadas durante o processo de visualização. Reduzindo essa quantidade de dados minimiza, por tanto, a quantidade de execução dessas operações. Assim, melhora a possibilidade de visualização da cena em tempo real.

4.2.2 Transmissão de dados gráficos

Em aplicações de ambientes virtuais baseados em Web a transmissão de dados gráficos é uma operação bastante crítica. Para visualizar a cena é necessário carregar os objetos que a compõe. Como a tendência atual é ter de maior fidelidade de representação em ambientes virtuais se faz necessário então uma maior quantidade de dados, podendo assim comprometer o tempo de carregar a cena já que a transmissão dos dados é mais demorada.

Um ambiente virtual pode ser formado por um objeto contínuo ou por múltiplos objetos separados (multi-arquivos). Ambientes formados por múltiplos arquivos também é outro fator que pode comprometer o desempenho de transmissão já que geralmente a transmissão de um único arquivo demora menos que a transmissão de múltiplos arquivos de tamanho total igual ao único arquivo.

Então, assim como durante a renderização, a redução de dados a serem transmitidos (para visualizar uma cena inicial do ambiente) diminuiria o tempo de carregamento de dados da cena. A transmissão dos demais dados restantes poderia ser feita de acordo com a necessidade deles.

4.3 Solução Proposta

Sabemos que a estrategia de dividir uma cena em regiões já foi proposta em múltiplos trabalhos. Foi pensando nisso que resolvemos revisitar essa abordagem ao propor dividir a cena de outra maneira. Nossa proposta se baseia em dividir a cena utilizando uma grade regular onde as primitivas da cena serão associadas ao bloco que a contém e a(s) face(s) (deste bloco) a qual ela está voltada. Posteriormente, a partir do frustum de visualização, é possível determinar quais faces dos blocos são visíveis e consequentemente pode-se gerar a lista de primitivas visíveis a partir desse conjunto de faces.

A Figura 4.2 ilustra nossa solução básica proposta para determinar o conjunto de primitivas potencialmente visíveis. Estamos primeiramente apresentando a solução básica, onde o vetor de visualização da câmera é paralela aos eixos de coordenadas, porque esse caso é mais simples para a formulação de transição bloco a bloco. A formulação da operação de transição bloco a bloco é demonstrada na Seção 5.2.2.

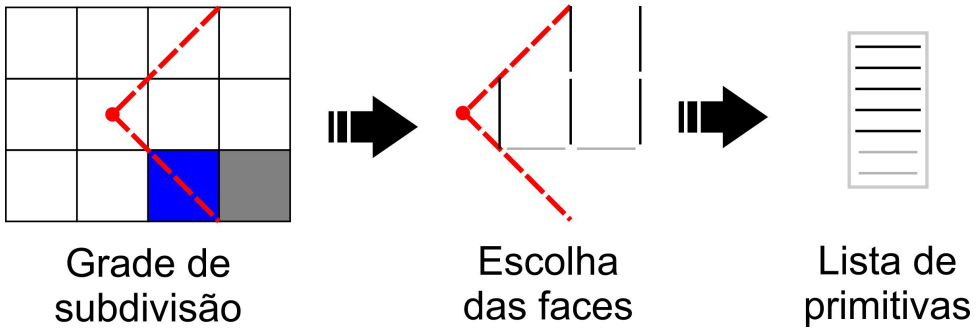


Figura 4.2: Ilustração da nossa solução básica proposta: subdividir a cena em blocos através de uma grade, escolher as faces dos blocos visíveis e gerar a lista das primitivas visíveis que estão contidas nas faces dos blocos.

Escolhemos a estrutura J_1^a como base para nossa grade regular pelas sua regras de transição de blocos e faces. Além disso, como proposta futura, desejamos aproveitar o fato da estrutura J_1^a ser adaptativa para refinar a determinação de oclusão em regiões críticas de cena.

4.3.1 Visualização não-paralela aos eixos de coordenadas

Apresentamos primeiramente a solução do caso básico pois, a solução para determinar o PVS quando a direção de visualização não está paralela aos eixos de coordenada é análoga. Neste caso, a determinação dos blocos visíveis é igual: se o bloco ou uma parte deste bloco está dentro do frustum de visualização então é considerado visível. Para determinação das faces dos blocos visíveis basta verificar qual face está voltada (ou parcialmente voltada) para a direção de visualização, se for o caso, essa face será marcada para gerar a lista das primitivas visíveis.

A Figura 4.3 ilustra a solução para a visualização não-paralela aos eixos de coordenadas.

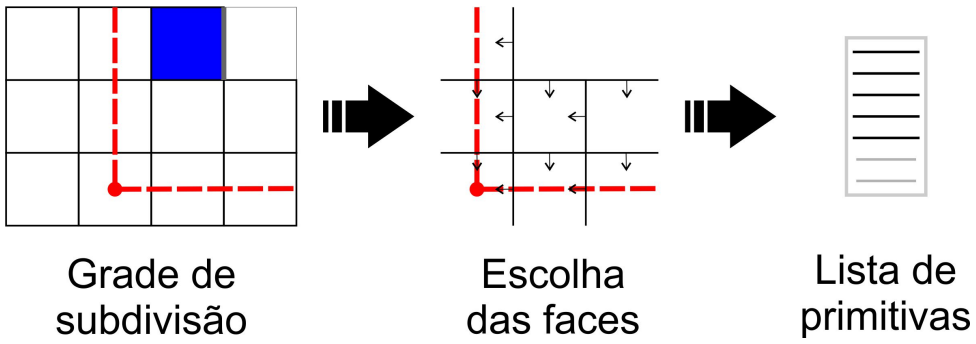


Figura 4.3: Ilustração da solução para a visualização não-paralela aos eixos de coordenadas: subdividir a cena em blocos através de uma grade, escolher as faces dos blocos visíveis (faces voltadas à direção de visualização) e gerar a lista das primitivas visíveis que estão contidas nas faces dos blocos.

4.3.2 Existência de primitivas em múltiplos blocos

O uso de blocos de subdivisão pode gerar a indexação de primitivas por mais de um bloco adjacentes, esse problema pode causar o processamento de uma mesma primitiva mais de uma vez. Para resolver esse problema, geramos uma rotina que identifica se uma dada primitiva já foi referenciada por um bloco anterior. Existe outra abordagem para solucionar esse problema, a de simplesmente forçar que uma primitiva só seja indexada por único bloco durante a etapa de subdivisão porém, nunca se pode garantir que esse bloco seja processado mesmo que seu adjacente seja. Essa abordagem pode gerar buracos já que primitivas (que preencheriam esse buraco) podem não ser acessadas. A Figura 4.4 ilustra esse problema.

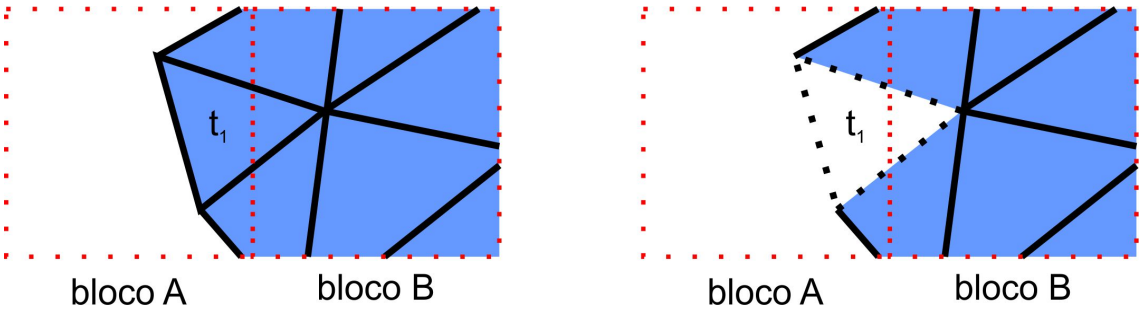


Figura 4.4: Ilustração do problema de indexação única de primitivas. No lado esquerdo, é ilustrado o caso em que o triângulo t_1 é visível no bloco B então deve ser processado. No lado direito, vemos que se o triângulo t_1 é somente indexado pelo bloco A e somente o bloco B é processado então t_1 não é processado, gerando, assim, um buraco na cena.

4.4 Visualização da Estrutura Cerebral

A fim de visualizar as estruturas de cerebrais a partir de dados DKI, é necessário resolver a sua equação tensorial. Os dados de entrada gerados por um exame DKI são: 5 valores de intensidade b (no nosso caso este valores são 500, 1000, 1500, 2000, 2500); 30 vetores direcionais de escaneamento do exame e; para cada valor de intensidade b e cada vetor direcional uma imagem volumétrica (conjunto de fatias) do cérebro. Em outras palavras, cada ponto (voxel) dessa palavra é representado por 150 valores. E é para cada um desses pontos que utilizaremos seus respectivos parâmetros para calcular: direcionamento, intensidade, tensor de difusão curtótica e curtose média. Para calcular esses parâmetros utilizamos o sistema de equações apresentado na subseção 2.6.1.

Para resolver o sistema de equação, utilizamos a biblioteca Eigen[1]. Esta biblioteca disponibiliza ferramentas capazes de resolver as equações tensoriais necessárias. E a partir dai obtemos os autovetores e autovalores de DT E KT.

Após os parâmetros serem obtidos, na próxima etapa, as malhas que representam as estruturas são geradas a partir de uma simples triangulação dos pontos obtidos. Posteriormente, utilizamos a estrutura de visualização para executar a etapa de pré-processamento antes da visualização.

Durante a visualização o usuário tem disponível uma interface para escolher quais componentes da estrutura do cérebro serão visualizadas a partir dos parâmetros desejados.

[1]Eigen (C++ library), site: http://eigen.tuxfamily.org - última vez acessado em 12/03/2014.

Capítulo 5

Implementação

Neste capítulo apresentamos o detalhamento da implementação das soluções propostas na Seção 4.3.

Baseamos a nossa estrutura de visualização na estrutura algébrica da J_1^a . Toda a cena 3D está contida dentro da grade criada pela estrutura J_1^a . Do ponto de vista da câmera, a estrutura calcula todos os elementos visíveis por ela. Então, toda vez que a câmera é movida, a operação de obtenção do conjunto visível de primitivas é refeita.

Nossa estrutura de visualização é capaz de determinar o *culling* de oclusão de duas maneiras: oclusão interna do bloco e oclusão entre blocos adjacentes. A existência de oclusão interna do bloco é identificada usando apenas as primitivas dentro de cada bloco e é feito para cada face do bloco, como ilustrado na Figura 5.1 para o rosto F_1. Para verificar a oclusão entre blocos adjacentes, a estrutura verifica se a face da frente (de acordo com a direção de visualização) está totalmente preenchida, se for o caso, fica evidente que a face do bloco adjacente atrás está totalmente bloqueada. Maiores detalhes sobre essa a determinação de oclusão serão apresentados na Seção 5.2.1.

A partir dos dados obtidos em exames de imageamento DKI, geramos uma série de informação referente as características da estrutura cerebral durante o dado exame. Para gerarmos essas informações, usamos a formulação matemática apresentada na Subseção 2.6.1. A implementação do método de geração de malhas a partir dos dados obtidos pela formulação de [Jensen et al. 2005] é apresentada de forma sistemática na Seção 5.3

A implementação da estrutura de visualização e de suas aplicações foi feita na linguagem de programação C++, para as operações de visualização utilizamos a biblioteca OpenGL. Para a aplicação de Visualização de Dados DKI utilizamos a biblioteca para gerar a janela do painel de controle utilizamos a biblioteca glUI[1] e para realizar os cálculos dos parâmetros DKI utilizamos a biblioteca Eigen. Todas as bibliotecas utilizadas são portáveis, ou seja, funcionam em qualquer sistema operacional, devido a isso, garantimos a portabilidade do nosso código.

[1]GLUI User Interface Library - Site: http://glui.sourceforge.net/ - última vez acessado em 12/03/2014.

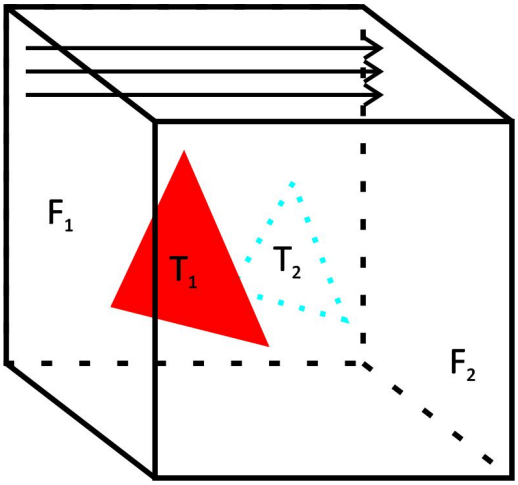


Figura 5.1: Ilustração da técnica de determinação de oclusão interna do bloco usando uma abordagem semelhante ao algoritmo de *Z-buffer*. Ao verificar o conjunto visível de primitivas para o rosto F_1, os raios determinam que a primitiva T_1 esconde a primitiva T_2. Vale lembrar que a técnica ilustrada verifica oclusão causada por grupos de primitivas e não somente primitivas individuais.

5.1 Fluxogramas

A Figura 5.2 ilustra a visão geral da estrutura de visualização. Basicamente, ele é subdividido em duas etapas: de pré-processamento e de visualização. Na Seção 5.2, estas etapas são melhor detalhadas.

5.1.1 Visão sistemática das aplicações

Detalhamos, a seguir, uma visão sistemática das aplicações que utilizam a estrutura de visualização.

1. Visualização com Percepção 3D - KIN3D
   Baseado no trabalho proposto por [Bista et al. 2013], implementamos uma aplicação capaz de gerar percepção 3D a partir do movimento da câmera. Devido ao constante movimento (tanto no movimento angular quanto no movimento cônico) da câmera a taxa de recálculo de conjunto de primitivas é elevada. Com isso em mente, acreditamos que esta aplicação é ideal para verificar a influência da taxa de recálculo sobre a taxa de quadros de visualização.
2. Visualização de Dados DKI
   A partir dos dados DKI é gerado um conjunto de malhas complexas que representam a estrutura cerebral. Por ser um modelo complexo e com alta quantidade de dados, utilizamos a estrutura de visualização para auxiliar a aplicação. Além disso, a aplicação é composta também controles interativos que possibilitam o usuário a escolher os objetos, cujos parâmetros são desejados, a serem visualizados.

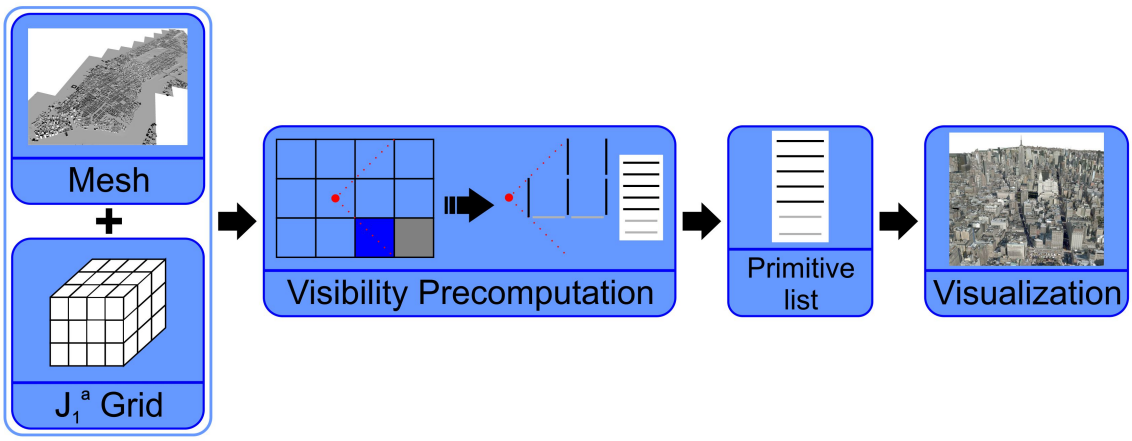


Figura 5.2: Visão geral da nossa estrutura de visualização. Nós ilustramos o estágio de pré-processamento onde, dadas as malhas que representam a cena e a grade básica da J_1^a , verificamos quais blocos dessa grade são visíveis e, a partir dai, determinamos quais faces de cada bloco são visíveis. Finalmente, obtemos a lista primitivas visível para serem visualizadas.

3. Visualização em Cavernas Virtuais
   Escolhemos implementar uma aplicação que simule um ambiente de caverna virtual por esse utilizar múltiplos pontos de projeção. Tendo em vista, nossa aplicação simula uma visualização da cena como se ela fosse vista por 3 câmeras distintas. Cada câmera tem então seu respectivo conjunto de primitivas potencialmente visível. A partir daí, todo o processo de interação e navegação é análogo ao caso da existência de uma só câmera visualizando a cena.

5.2 Etapas da Estrutura de Visualização

Detalhamos aqui cada uma das etapas de execução da estrutura de visualização.

5.2.1 Etapa de Preprocessamento

Durante a etapa de pré-processamento, usamos uma grade baseada na estrutura básica da J_1^a para gerar os blocos de visualização.

Esta etapa é executada em quatro passos:

1. Dado, como entrada do usuário, um valor em que está estabelecida a dimensão mínima da grade ao longo de um de seus eixos, podemos determinar o tamanho das arestas e calcular o resto da dimensão da grade.
2. Cada triângulo é primeiramente mapeado ao(s) bloco(s) da grade onde está contido;
3. Em seguida, usa-se as normais do triângulo para identificar para quais faces do bloco o triângulo está voltado. Nesse passo, tratamos a determinação de *back-face culling*;

4. A oclusão interna de cada bloco é então calculada usando uma abordagem semelhante ao algoritmo de *Z-buffer*. Uma lista final de primitivas é atribuída a cada face dos blocos. Durante este passo, descobre-se se cada uma das faces dos blocos é totalmente preenchida ou não, isto é necessário para determinar a oclusão entre blocos adjacentes na fase de visualização. Esta etapa é ilustrada na Figura 5.1.

5.2.2 Etapa de Visualização

Usamos a grade para verificar quais as primitivas da cena que estão sendo observadas diretamente. Dada a posição da câmera e seu vetor de direcionamento, usamos a função de transição de estrutura J_1^a para acessar os blocos ao longo da linha de visão. O vetor de direcionamento também é usado para determinar quais as faces do bloco serão usadas para compor a lista final de primitivas potencialmente visíveis.

Para obter a lista de primitivas visíveis é necessário acessar cada um dos blocos dentro do volume de visualização, para isso utilizamos uma operação de transição algébrica cuja transição tem como ponto de partida o bloco onde a câmera esta localizada. Todas as primitivas do bloco inicial (i, j) são carregadas para visualização, e a partir desse bloco são feitas as transições para os demais blocos (ilustrado para o caso 2D na Figura 5.3). Nesta figura, vemos que o próximo bloco a ser acessado é o bloco $(i+1, j)$, as primitivas referenciadas pela face voltada para (i, j) são então adicionadas a lista de primitivas. Além disso as faces adjacentes dos blocos $(i+1, j+1)$ e $(i+1, j-1)$ também tem suas primitivas adicionadas a lista de primitivas. A transição continua para os blocos $(i+2, j+1)$, $(i+2, j)$ e $(i+2, j-1)$ onde suas faces voltadas para (i, j) tem suas primitivas adicionadas à lista de primitivas e, o mesmo ocorre para as primitivas das faces de $(i+2, j+2)$ e $(i+2, j-2)$ adjacentes aos blocos $(i+2, j+1)$ e $(i+2, j-1)$ respectivamente. A operação de transição finaliza quando se chega ao fim da grade.

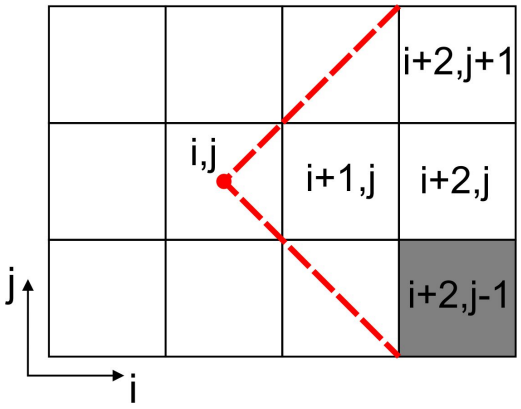


Figura 5.3: Ilustração da operação de transição de blocos, apresentamos uma representação 2D análoga para o caso 3D. Neste exemplo, cada bloco visível é visitado para verificação de face visível. A face visível do bloco $(i+2, j-1)$ está totalmente preenchida então os demais blocos seguintes adjacentes a esse bloco são bloqueados.

Antes de cada transição de bloco a bloco, é verificado se a face está totalmente preenchida, a fim de permitir esta transição. Se for o caso, isto significa que o conjunto

visível da face verificada oculta completamente o conjunto visível da face do bloco adjacente, de modo que o bloco escondido e seus subsequentes blocos adjacentes não serão necessários para o estágio de renderização. Como ilustra a Figura 5.3, a face do bloco cinza $(i+2, j-1)$ está totalmente preenchida então os demais blocos atrás deste bloco são bloqueados.

A lista final de primitivas (visíveis) dos blocos será então usada para compor o conjunto de primitivas potencialmente visíveis usado no estágio de renderização. Como foi dito anteriormente, cada vez que a câmera é movida ou redirecionada, o cálculo para obter o conjunto visível de primitivas é refeito.

A transição bloco a bloco para a visualização não-paralela aos eixos de coordenadas se difere do caso básico porém, a transição também segue a direção de visualização ao selecionar os blocos opostos as faces consideradas visíveis do bloco atual.

Nosso método é capaz de lidar bem com a existência de transparência em primitivas, para ambos os casos determinação de oclusão nos blocos. Esse tratamento é feito durante a fase verificação de oclusão interna do bloco. Quando o raio encontra uma primitiva transparente/semitransparente, esta primitiva será marcada para processamento de transparência e o raio vai continuar a tentar e encontrar uma outra primitiva ao longo de seu caminho dentro do respectivo bloco.

Vale salientar que utilizamos funções básicas de OpenGl para gerar a visualização. Não utilizamos funcionalidade *glList* pois o seu uso afetaria positivamente os resultados para o caso de uso dos modelos completos e tem potencial de afetar negativamente com o uso de PVS já que quando se navega uma cena a lista de primitivas é atualizada.

5.3 Geração e Visualização de Dados Médicos

Essa aplicação é executada em duas etapas distintas: geração de malhas das estruturas cerebrais e visualização dessas malhas geradas.

Durante a etapa de geração de malha, utilizamos a biblioteca Eigen para calcular o direcionamento, a intensidade e o tensor de difusão curtótica. Essa é uma etapa demorada devido ao fato de que para cada ponto das imagens volumétricas geradas pelo exame DKI são atribuidos 150 valores de intensidade de acordo com o seu respectivo vetor de direcionamento de escaneamento e intensidade do valor b. A Figura 5.4 ilustra a visualização transversal de cinco imagens volumétricas de entrada cujos valor b=2000 e somente os vetores de direcionamento são diferentes. Então sobre cada ponto é necessário resolver um sistema complexo para obter os dados desejados para a geração de malhas (para obter o valor a curtose média).

Posteriormente utilizamos uma simples triangulação dos valores de MK para gerar as malhas correspondentes aos parâmetros obtidos. A etapa de visualização é a mesma do processo executado pela estrutura de visualização. A única diferença é a presença de controles que permitem selecionar regiões e parâmetros desejados as serem visualizados.

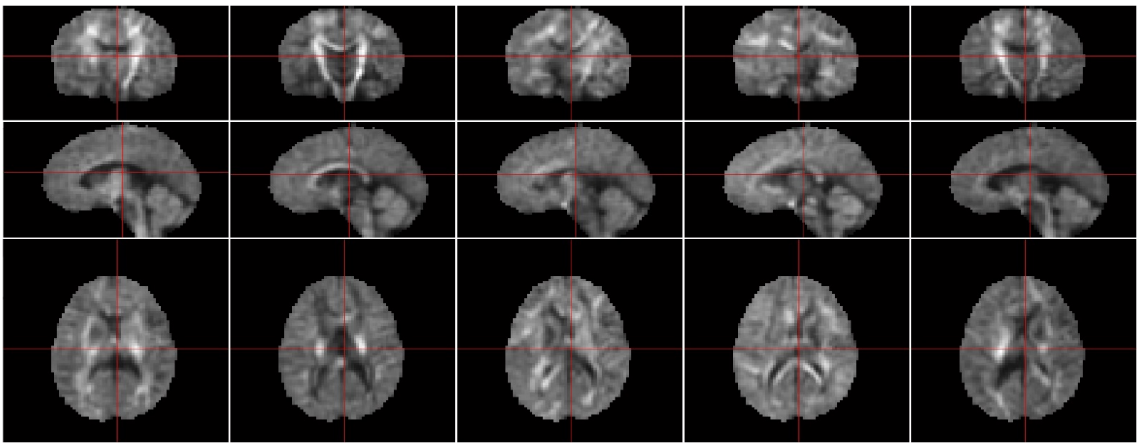


Figura 5.4: Visualização transversal de cinco imagens volumétricas cerebrais de entrada. Para essas imagens o valor b=2000 e cada uma tem sua respectivo vetor de direcionamento. Note que essas imagens comprovam que diferentes direções de obtenção do sinal DKI geram diferentes valores devido à direção de difusidade da água ser diferente nas variadas regiões do cérebro.

Capítulo 6

Experimentos e Resultados

Os experimentos apresentados na Seção 6.1 são realizados em uma aplicação de visualização a partir do ponto de vista (POV) da câmera. Para avaliar a eficiência da nossa estrutura de visualização esses experimentos realizados, onde cada cena/modelo é visualizado individualmente e uma sequência de deslocamentos da câmera é realizada para navegar ao longo da cena. Após cada deslocamento serão verificados uma série de informações que caracterizam a eficiência dessa estrutura.

Para este experimento, utilizamos uma série de modelos de malha obtidos a partir do repositório *Aim at-Shapes*: Leão Chinês, Vaso, Tatu, Mão e Eros . A Figura 6 ilustra cada modelo. As características da forma e o alto nível de detalhe dos modelos tornam os ideais para testar a eficiência da nossa estrutura para a visualização de malhas individuais. Nesse experimento, também foi utilizado o modelo da ilha de Manhattan (ilustrado na Figura 6.2) passos de navegação foram realizados ao longo do modelo a partir de variados pontos de vista. Este modelo é constituído por um conjunto de 306 malhas totalizando 3 milhões de polígonos, 5 milhões de vértices e 296 imagens de texturas cada uma com resolução de 4096x4096.

As Figuras 6.3 e 6.4 ilustram as sequências de navagação utilizadas nos testes. Para os testes com somente um objeto gráfico utlizamos os mesmo passos ilustrados na Figura 6.3. A Figura 6.5 ilustra um dos passos de navegação do modelo Manhattan de forma mais detalhada.

Realizamos os experimentos em um PC Intel Core i7 2.00GHz com 8GB de RAM, com uma placa gráfica Radeon HD 6770Ms e em execução no sistema operacional Windows 7 (64bits). Quanto a escolha do sistema operacional, vale ressaltar que o nosso código é portável, escolhemos tal sistema para realizarmos nosso teste simplesmente pela existência de ferramentas que auxiliaram no desenvolvimento do código e na verificação e comparação de resultados.

Além dos experimentos, comparamos a eficiência de nossso método em relação à eficiência de outros métodos e foram também implementadas três aplicações onde a estrutura de visualização é utilizada. Essas aplicações estão apresentadas nas Seção 6.6.

6.1 Setup Experimental

Durante os experimentos, verificamos os seguintes resultados obtidos:

Figura 6.1: Modelos utilizados para testar o nosso método. Todos esses modelos são interessantes devido a suas formas e alto nível de detalhe. Malhas Leão Chinês, Vaso, Tatu, Mão e Eros são respectivamente compostos por 108k, 113k, 344k, 391k e 395k triângulos.

1. A média da taxa de quadros durante a visualização (cena estática $T\bar{Q}_S$ e navegando a cena $T\bar{Q}_N$) em comparação com a utilização de todas as primitivas da cena $T\bar{Q}_{PRI}$. Essa é uma medida básica para qualquer aplicação interativa 3D e é medida em quadros por segundo;
2. A razão média ($\bar{R}_{PVS/PRI}$) entre o conjunto de primitivas potencialmente visíveis e o total de primitivas que compõem a cena (#Pri). Também analisamos a razão média ao se utilizar somente *view-frustrum* e *back-face culling* também com relação a #Pri ($\bar{R}_{VF+BF/PRI}$), verificamos isso para identificar a proporção de eliminação de cada técnica de *culling*. A razão é medida entre 0 e 1 e pode ser interpretada como por exemplo: se a razão é igual a 0.10 então isso implica que o método visualiza somente 10% (eliminando os outros 90%) das primitivas originais da cena;
3. Razão média de taxa de superestimação (proposta por [Cohen-Or et al. 2003]), onde comparamos a razão entre a quantidade real de primitivas visíveis (VS) e o tamanho do conjunto de primitivas potencialmente visíveis (PVS), em outras palavras, a

Figura 6.2: Ilustração do modelo da ilha de Manhattan. O modelo composto por vários objetos tem potencial para ser ideal nos experimentos de detecção de oclusão. Este modelo é composto por por um conjunto de 306 malhas totalizando 3 milhões de polígonos, 5 milhões de vértices e 296 imagens de texturas em alta resolução.

razão é igual a VS/PVS.

Assim como os dados resultantes de 1, calculamos o valor médio dos dados em 2 e 3 após a navegação de cada cena. Para cada resultado, também apresentamos seu desvio padrão (σ). A fim de proporcionar uma comparação justa entre cada valor de razão, a mesma sequência das etapas de navegação de cena são feitas para cada experimento executado em cada modelo.

Nas seções a seguir, as Tabelas 6.1 e 6.2 apresentam os dados obtidos durante os experimentos. Como nós já relatamos anteriormente, os experimentos realizados em cada um dos modelos foram feitos usando a mesma sequência de passos de navegação e mesma dimensão grade.

6.2 Experimentos de Tempo

Como podemos observar na tabela 6.1, a taxa de quadros obtidas quando visualizando cada modelo utilizando a estrutura de visualização é melhor que utilizar todo o dado do

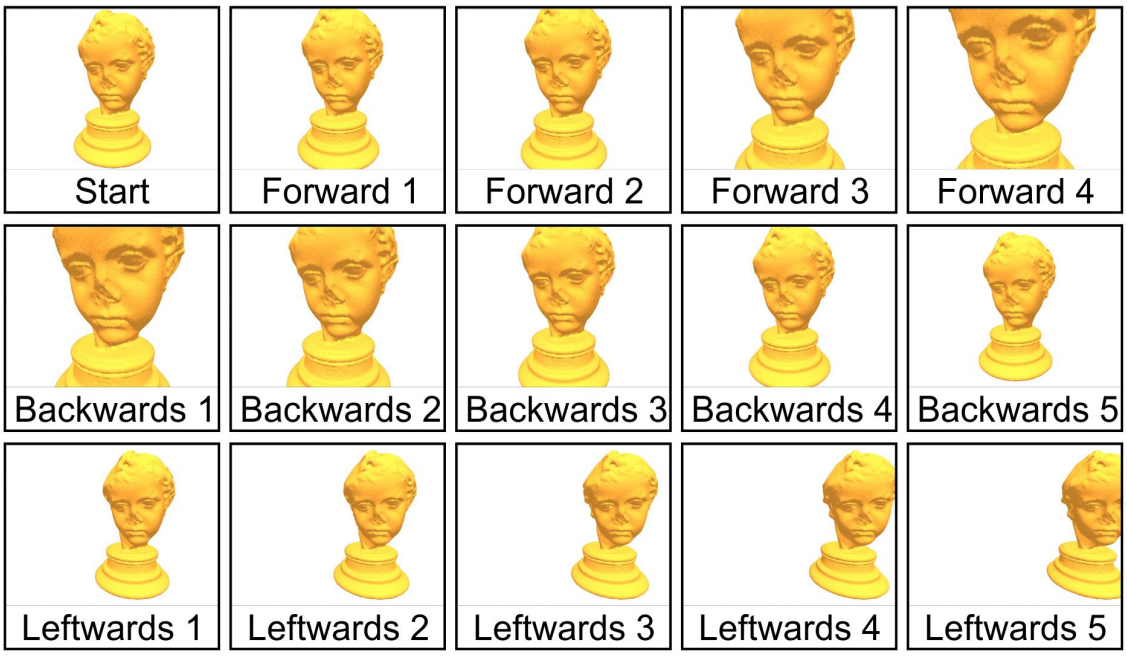


Figura 6.3: Ilustração do conjunto de amostras da cena obtidas durante a sequência de navegação. Primeiramente, a câmera se move na direção do objeto (frente), depois se afasta desse objeto (para trás) e finalmente se movimenta para a esquerda.

modelo. Esse resultado confirma que a estrutura está funcionando corretamente, o resultado mais importante da serie de experimentos é se o recálculo do conjunto de primitivas visíveis afeta a taxa de quadros. Ao compararmos as colunas $T\bar{Q}_N$ e $T\bar{Q}_s$, podemos ver que essa etapa de recálculo executada a cada passo da sequência de navegação não afeta tanto a taxa de quadros, diminuindo a no máxima em 15%.

Podemos notar que, no caso do modelo Manhattan, a taxa de quadros teve uma melhora considerável. Veremos na Seção 6.3 que para esse modelo a redução de dados foi significante, melhorando assim a taxa de quadros.

6.3 Conjunto de Primitivas Potencialmente Visíveis

Na Tabela 6.2, podemos ver que a redução dos dados a serem utilizados para a fase visualizações é bastante significativa. Embora o *back-face* e *view-frustrum culling* fizeram a maior parte da remoção de primitivas escondidas, a remoção por oclusão ainda removeu uma quantidade razoável de primitivas escondidas. E, no caso do modelo do vaso, devido à sua forma côncava, podemos ver que a remoção por oclusão reduziu uma proporção mais elevada de primitivas (removeu $29,1\%$ em comparação com os outros modelos simples foram inferiores a 22%).

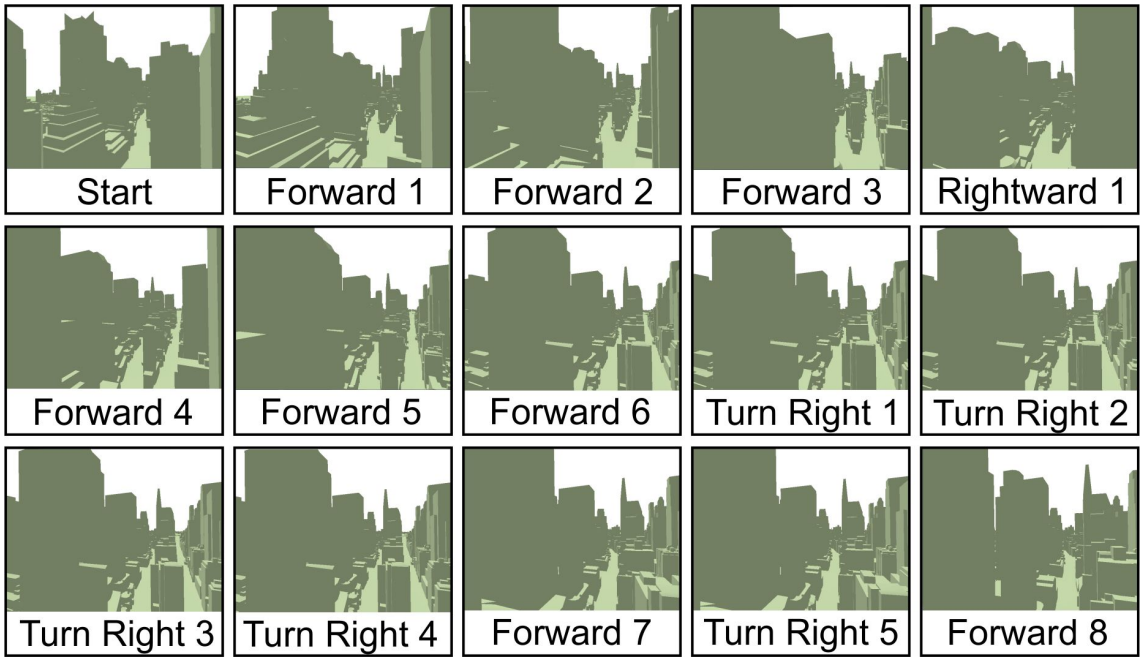


Figura 6.4: Ilustração do conjunto de amostras da cena obtidas durante a sequência de navegação. Os passos de navegação são realizados pelas operações de mover para frente, mover para a direita e virar para a direita.

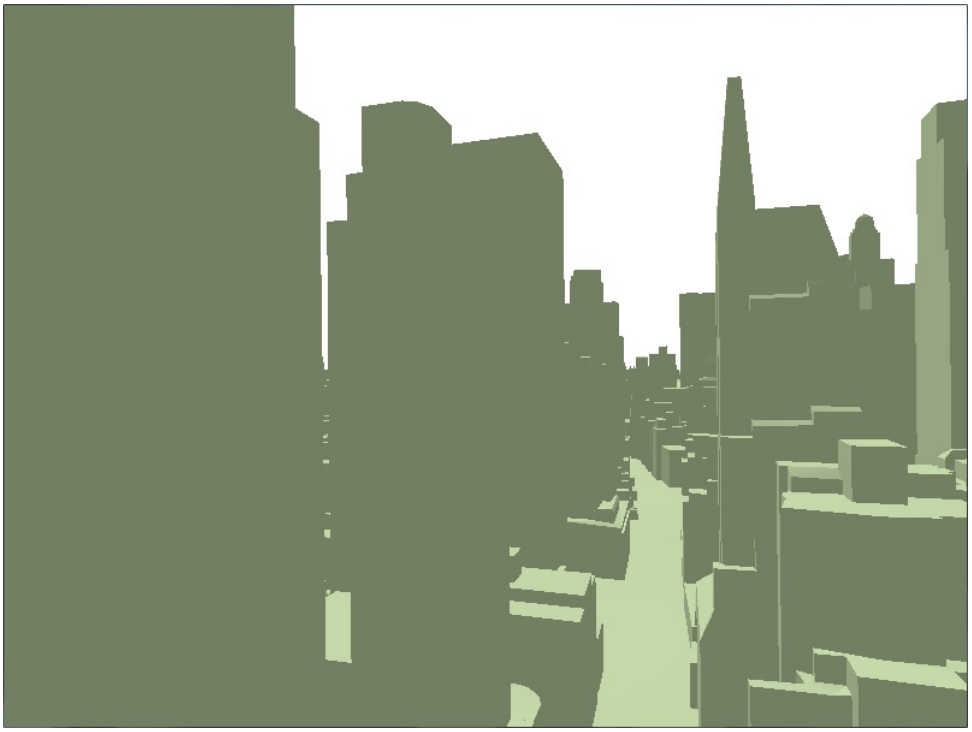


Figura 6.5: Visualização do modelo da ilha de Manhattan a partir de um dos passos de navegação. Verificamos nessa cena também a possibilidade de haver buracos na cena.

Tabela 6.1: Comparação dos resultados obtidos nos experimentos de comparação de taxa de quadro para cada modelo. Analisamos a taxa de quadro de uma cena estática $T\bar{Q}_S$, quando navegando em uma cena $T\bar{Q}_N$ e de uma cena composta por todos os triângulos $T\bar{Q}_{PRI}$. Além da média, apresentamos o desvio padrão de cada experimento.

Modelo	$T\bar{Q}_s$	$\sigma(T\bar{Q}_s)$	$T\bar{Q}_N$	$\sigma(T\bar{Q}_N)$	$T\bar{Q}_{PRI}$	$\sigma(T\bar{Q}_{PRI})$
Leão Chinês	28.569	0.967	25.773	2.591	18.883	2.766
Vaso	26.842	1.304	24.962	2.215	19.890	0.924
Tatu	11.367	0.557	9.756	0.860	6.221	0.245
Mão	7.161	0.410	6.929	0.638	5.628	0.223
Eros	6.957	0.456	6.566	0.703	5.364	0.326
Manhattan	10.75	0.884	9.85	0.788	0.6	0.043

Tabela 6.2: Comparação entre a razão média (e do desvio padrão) do conjunto onde as primitivas são removidas pelo *view-frustum* e *backface culling* com relação ao #*Pri* ($\bar{R}_{VF+BF/PRI}$) e a razão média (e do desvio padrão) do PVS com relação ao #*Pri* ($\bar{R}_{PVS/PRI}$). Analisamos também a razão entre a quantidade de primitivas realmente visíveis e o tamanho do conjunto das primitivas potencialmente visíveis do experimento ($\sigma(\bar{R}_{VS/PVS})$).

Modelo	#Pri	$\bar{R}_{VF+BF/PRI}$	$\sigma(\bar{R}_{VF+BF/PRI})$	$\bar{R}_{PVS/PRI}$	$\sigma(\bar{R}_{VS/PVS})$	VS/PVS	$\sigma(VS/PVS)$
Leão Chinês	108k	0.459	0.005	0.111	0.001	0.913	0.058
Vaso	113k	0.452	0.002	0.291	0.003	0.957	0.032
Tatu	344k	0.411	0.004	0.165	0.002	0.894	0.075
Mão	391k	0.467	0.002	0.132	0.003	0.914	0.067
Eros	395k	0.401	0.002	0.215	0.003	0.856	0.077
Manhattan	3000k	0.726	0.015	0.204	0.031	0.821	0.061

Quanto ao modelo de Manhattan, vimos que a proporção de remoção de primitivas foi semelhante aos outros modelos. Porém, quando analisamos quanto ao valor numérico de quantidade de primitivas removidas, identificamos a razão da melhora da taxa de quadros. Nesse caso, a quantidade de primitivas processadas foram reduzidas de 3000k para em média 210k.

6.4 Análise da Estrutura

Além das análises numéricas, detalhadas nas seções anteriores, quanto ao desempenho da estrutura de visualização. Analisamos também a o desempenho quanto a qualidade visual de dados (Seção 6.4.1) e também quanto ao uso dessa estrutura para transmissão de dados 3D (Seção 6.4.2).

6.4.1 Qualidade Visual dos Dados

Além da importância de testarmos parâmetros de eficiência da estrutura, é importante também testarmos a qualidade visual dos objetos gráficos e das cenas. Esses resultados são ilustrados nas Figuras 6, 6.2, 6.6 e 6.5. Durante esse teste tentamos verificar visualmente se há buracos na cena, ou seja, se alguma primitiva que deveria ser visualizada está ausente.

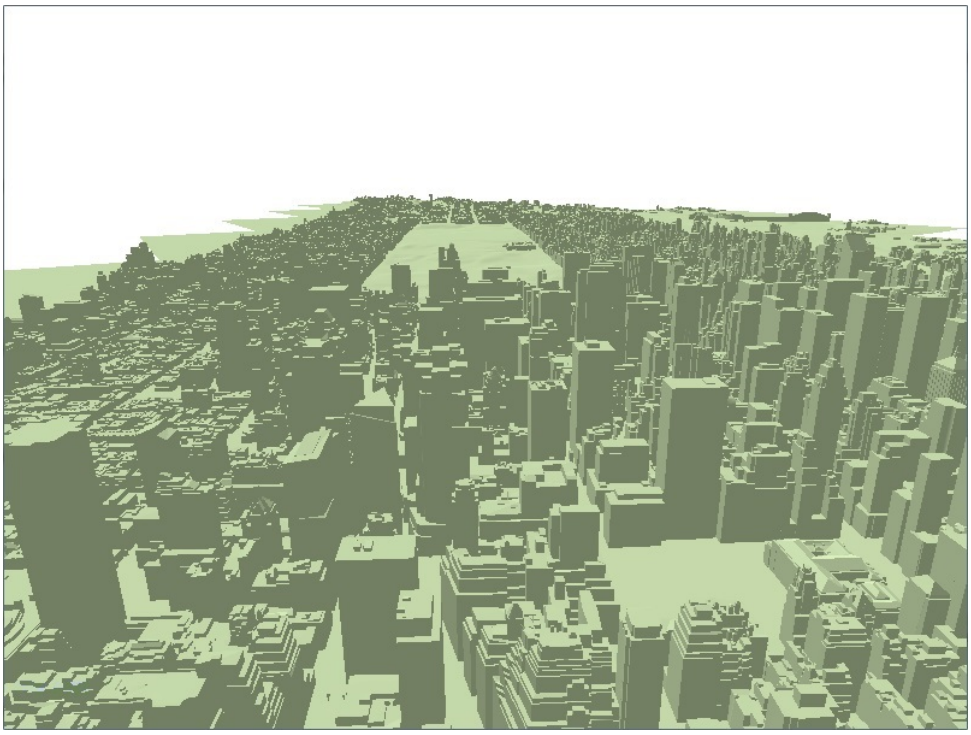


Figura 6.6: Visualização de outra região do modelo da ilha de Manhattan. A verificação de qualidade visual é importante especialmente para determinar a existência ou não de buracos na cena.

Tabela 6.3: Comparação de tempo de transmissão em segundos para três taxas de download (i) 56 Kb, (ii) 780 Kb e (iii) 1 Gb. São analisados os tempos de carregamento tanto para o conjunto completo de primitivas originais (#*Pri*) quanto para carregar o tamanho do conjunto de primitivas potencialmente visíveis (PVS).

			56Kb		780Kb		1Gb	
Modelo	#Pri	PVS	Tempo - #Pri	Tempo - PVS	Tempo - #Pri	Tempo - PVS	Tempo - #Pri	Tempo - PVS
Leão Chinês	108k	46k	60.3	25.9	4.8	2.1	0.003	0.001
Vaso	113k	29k	63.1	16.2	5.0	1.3	0.003	0.001
Tatu	344k	146k	192.0	81.4	15.2	6.4	0.01	0.004
Mão	391k	157k	218.2	87.5	17.3	6.9	0.01	0.005
Eros	395k	152k	220.4	84.6	17.4	6.7	0.01	0.005
Manhattan	3000k	210k	1674.1	117.2	132.4	9.3	0.09	0.006

6.4.2 Análise de uso em transmissão de dados 3D

Em aplicações de ambientes virtuais baseados em *web*, é importante analisar o desempenho da transmissão dos dados que compõem a cena do ambiente. Apresentamos na Tabela 6.3 uma análise numérica da transmissão de dados nas taxas de 56 Kb (de um modem), 780 Kb (DSL) e 1 Gb (fibra ótica). Para cada taxa comparamos o tempo de download tanto para carregar o conjunto completo de primitivas originais (Tempo - #Pri) quanto para carregar o tamanho do conjunto de primitivas potencialmente visíveis (Tempo - PVS).

Vale lembrar que, na Tabela 6.3 (assim como nas tabelas 6.1 e 6.2), só estamos analisando valores relacionados às primitivas (triângulos). Durante a transmissão de dados gráficos informações como coordenadas de pontos, vetor das normais, vetor de textura e as imagens de texturas também é crítica para o desempenho desta operação. Porém, como o nosso foco principal é a redução de primitivas, resolvemos não analisar redução desses dados mesmo que estes sejam também bastante reduzidos.

6.5 Comparação com Trabalhos Relacionados

Comparamos os nossos resultados obtidos nos testes com outros métodos existentes. Para ser uma comparação justa, testamos todos esses métodos com os mesmos modelos e passos de navegação utilizados no teste do nosso método. Os métodos escolhidos para comparação são os trabalhos de Cohen-Or et al. [Cohen-Or et al. 1998], Schaufler et al. [Schaufler et al. 2000] e, Durand et al. [Durand et al. 2000]. Esse métodos são conservativos, lidam com toda a cena 3D para determinação de visibilidade e calculam o PVS durante a etapa de precomputação, além disso, os oclusores são tratados de maneira semelhante ao nosso método. A seguir compararemos um a um cada método com relação ao nosso método.

- Cohen-Or et al. - Devido ao fato de trabalhar com oclusores separadamente, este método elimina uma quantidade significativamente menor de primitivas ocultas. No caso do modelo Manhattan isso é evidente pois é composto por múltiplos objetos. A proporção de remoção de primitivas ocultas foi em torno de 15% menor em comparação ao nosso método.
- Schaufler et al. - Este método trata visibilidade a partir de células de visualização. Objetos oclusores são extendidos seguindo a direção de visualização da câmera englobando assim todas as primitivas escondidas por ele. Quando comparado com o nosso método, verificamos que ambos tem resultados semelhante quanto a remoção de primitivas ocultas, em certos casos, notou-se que um método é mais eficiente e, em outros casos, o outro método é mais eficiente.
- Durand et al. - Neste método primitivas oclusoras e possivelmente ocluidas são projetadas em um plano, a primitiva ocluida é dita escondida se sua projeção é totalmente coberta por projeções cumulativas de oclusores. Este método é bastante eficiente ao tratar oclusão entre objetos, porém não verificamos nenhum caso onde ele trata oclusões de regiões de um objeto causadas por outras regiões desse objeto.

Mesmo assim, para os casos testados, os resultados dos testes deste método em comparação com o nosso método são bastante semelhantes.

6.6 Aplicações

Além de testarmos a eficiência da estrutura de visualização nos experimentos de desempenho apresentados nesse capítulo, aplicamos a estrutura em 3 aplicações: visualização com percepção 3D, visualização em cavernas virtuais e visualização de dados DKI. A seguir, iremos detalhar cada uma dessas aplicações.

6.6.1 Visualização com Percepção 3D

Aplicamos a estrutura em conjunto com a técnica desenvolvida por Bista et al. **??** para implementar um visualizador com percepção 3D. Para isto, o ponto de vista (da câmera) é deslocado em uma série de movimentos repetitivos ilustrados na Figura 6.7. Como a Figura apresenta, são dois os tipos de movimentos: movimento angular onde a câmera é rotacionada no seu eixo do vetor *up* e movimento em forma de um pêndulo cônico.

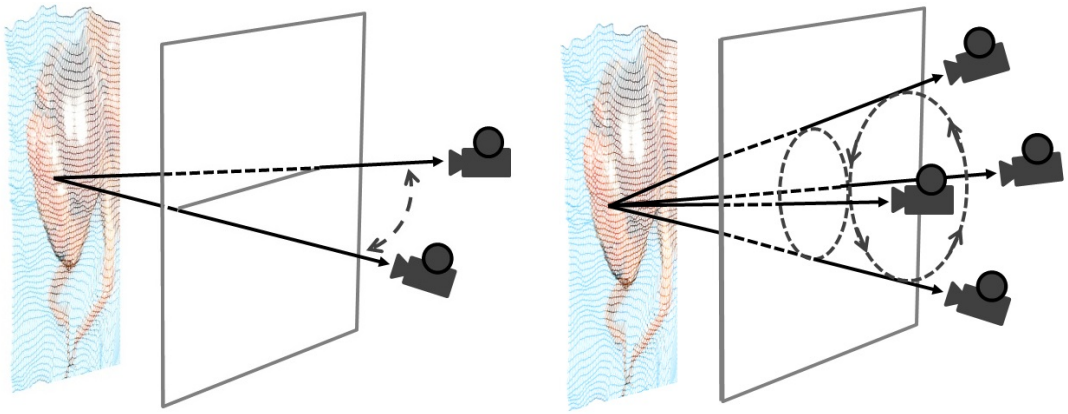


Figura 6.7: Ilustração dos modelos de deslocamento de câmera para gerar percepção 3D. A esquerda movimento angular onde a câmera é rotacionada em relação a um derterminado ponto da cena, e a direita, câmera é deslocada em movimento em forma de um pêndulo cônico.

Essa técnica foi desenvolvida baseado no método de *Kinetic Depth Effect* (Efeito Profundidade causado por cinética). O seu objetivo principal foi de disponibilizar um meio de percepção 3D sem ter o problema de enjoo ou dor de cabeça causado por dispositivos atuais de visualização 3D [Bista et al. 2013]. Durante os experimentos desta técnica com vários usuários foi comprovado que tais movimentos são capazes de gerar a percepção 3D sem causar tais problemas quando este visualizavam a cena. Foi comprovado também que o movimento cônico é capaz de gerar melhor percepção que o movimento angular.

6.6.2 Visualização em Cavernas Virtuais

Propomos também uma aplicação baseada em um ambiente de caverna virtual, onde a cena visualizada é projetada usando mais de um projetor (mais de um ponto de vista diferente). Queremos demonstrar que nosso método também é eficiente para esse tipo de aplicação.

Durante a execução dessa operação verificamos que a quantidade dados, para uma simulação com três câmeras, aumentou em média de 3.4 vezes com relação ao tamanho médio obtido durante a execução sequência de navegação ilustrada na Figura 6.4. Esse aumento era esperado já que triplicamos a quantidade de câmeras. Outro fato que vale a pena discutir é que como o "aparente volume de visualização" aumentou (diminuindo assim a proporção da eliminação a partir do volume de visualização), a proporção de remoção de primitivas ocultas também aumentou.

A Figura 6.8 ilustra a visualização de uma cena de Manhattan a partir de três pontos de vista.

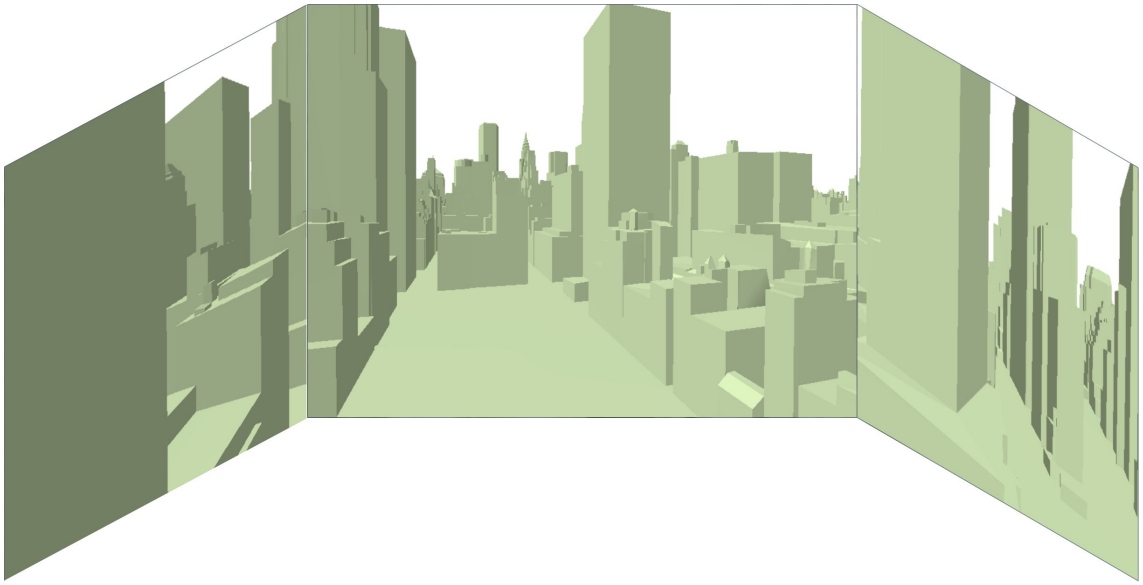


Figura 6.8: Ilustração da visualização de uma cena de Manhattan a partir de três pontos de vista (simulando assim como seria em uma caverna virtua).

6.6.3 Visualização de Dados DKI

Após as malhas que representam a estrutura cerebral forem geradas, a aplicação de visualização de dados DKI visualiza essa malhas de acordo com os parâmetros selecionados pelo usuário no painel de controle. Vale lembrar que, devido a complexidade e tamanho de malhas geradas, utilizamos nossa estrutura de visualização nessa aplicação.

A Figura 6.9 ilustra os valores do autovetor (V_1, V_2, V_3) e a curtose média obtidos para os dados detalhados na Seção 5.3. Utilizamos os autovetores para determinar o direcionamento de fibras (cujo resultado é ilustrado na Figura 6.10) e a curtose média para geração de malha 3D (ilustrado na Figura 6.11).

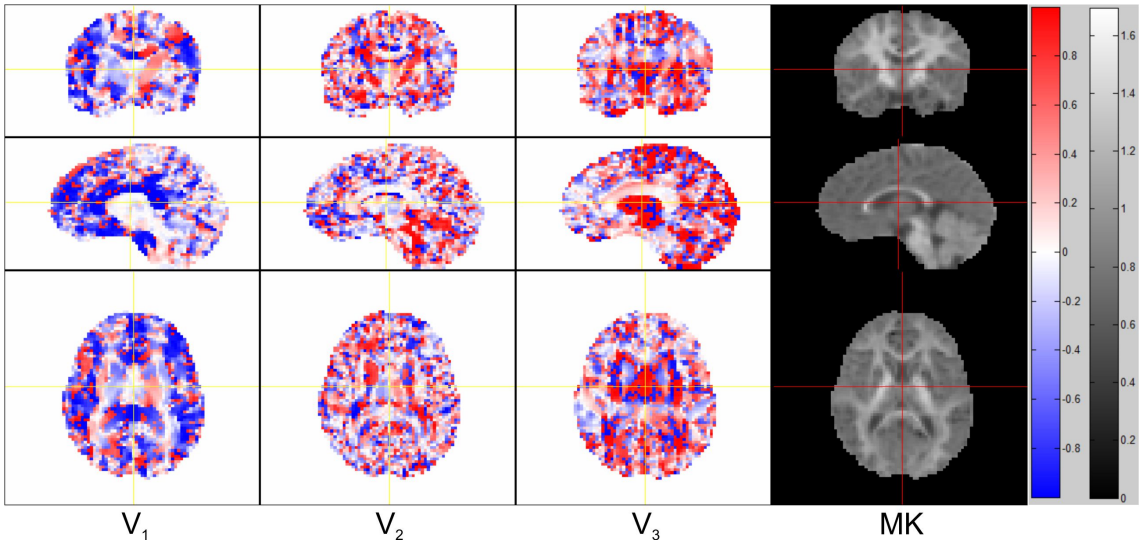


Figura 6.9: Ilustração dos valores do autovetor (V_1, V_2, V_3) e da curtose média (MK) com suas respectivas escalas de valores.

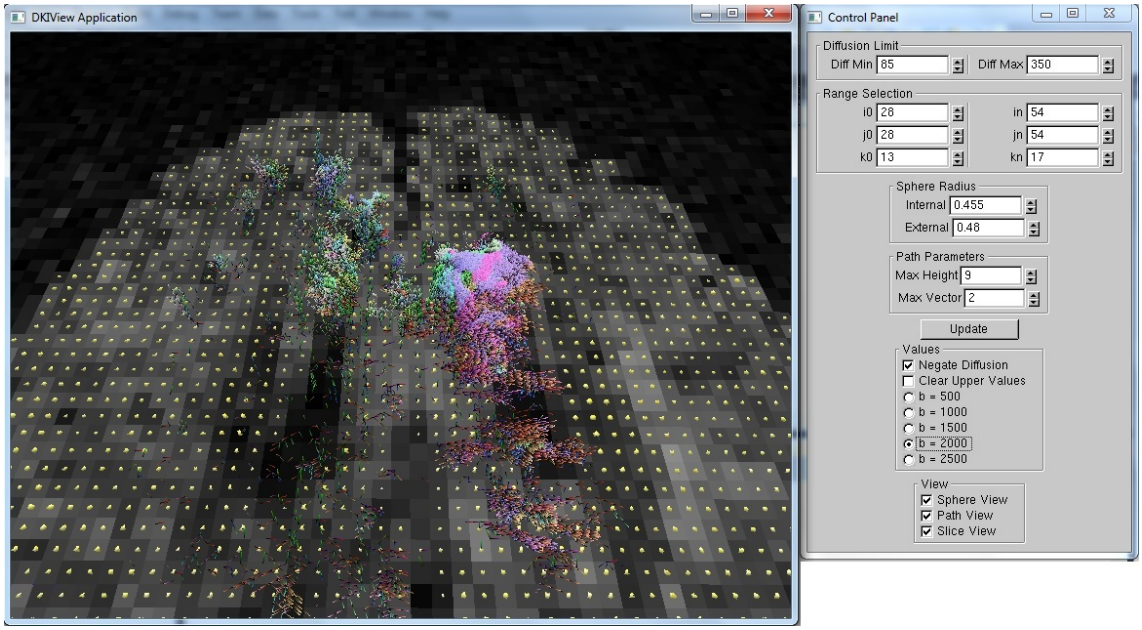


Figura 6.10: Ilustração da aplicação de visualização de dados DKI: a esquerda a janela de visualização e a direita o painel de controle. Nessa aplicação visualizamos ao mesmo tempo, a partir de uma fatia da imagem volumétrica, o traçado das fibras de massa branca, o valor dos sinais DKI (das 30 direções e para o valor b=2000) dos nodos da fatia em questão e a textura de uma das 150 imagens da fatia.

A aplicação de visualização é composta por duas janelas: janela de visualização e painel de controle. Como ilustra a Figura 6.10 que representa a visualização do valor do sinal DKI e traçado das fibras. Pelo painel de controle podemos selecionar uma re-

gião para geração do traçado, especificar os tamanhos mínimo e máximo das esferas para gerar o tracejado, especificar a altura máxima do tracejado (que inicializou a partir da fatia), escolher qual das texturas da fatia será utilizada assim como qual valor é utilizado para representar o valor dos sinais DKI nas 30 direções e, finalmente, selecionar quais as estruturas que devem ser visualizadas.

As Figuras 6.11 e 6.12 ilustram a visualização da estrutura a partir dos valores de curtose média, a primeira figura ilustra todo o cérebro. E a segunda ilustra uma região do cérebro a partir da seleção de intervalo dos valores de curtose média para a serem visualizados.

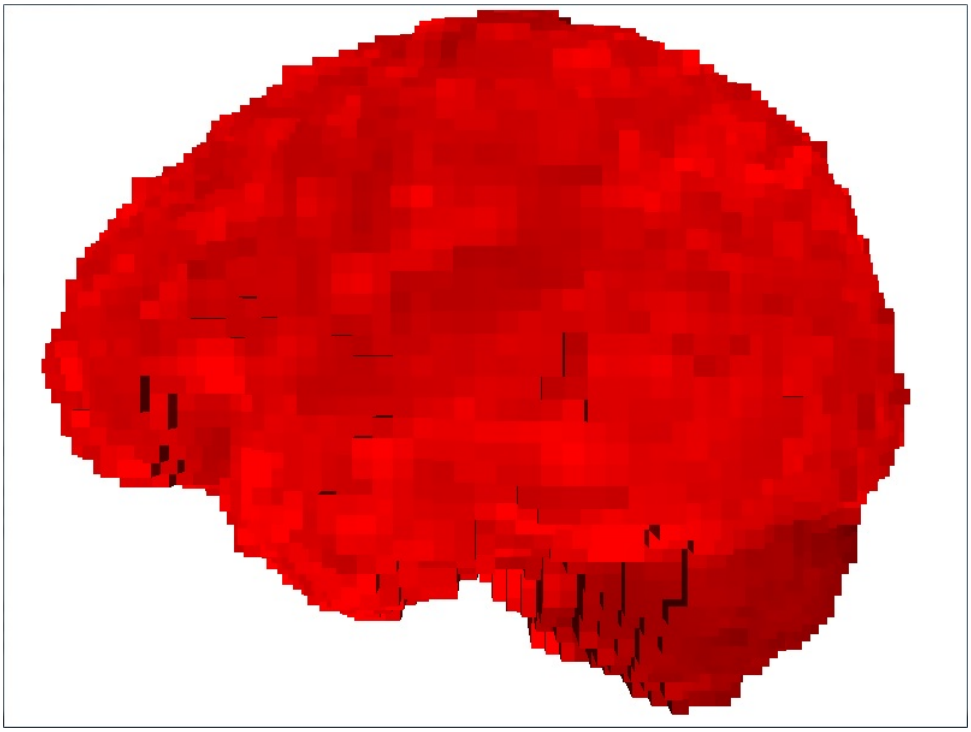


Figura 6.11: .Visualização de todo o cérebro a partir do parâmetro MK.

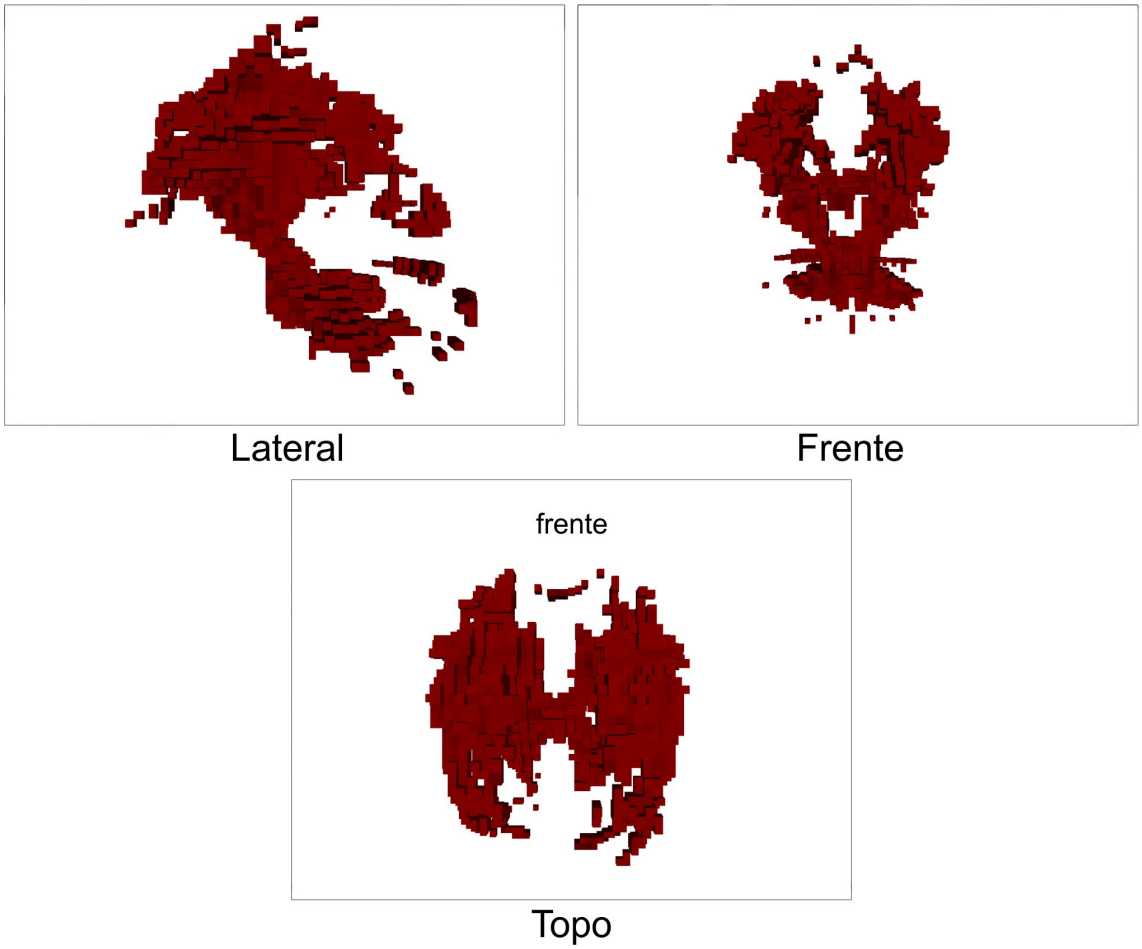


Figura 6.12: .Visualização de uma região do cérebro a partir da seleção de um intervalo de valores do parâmetro MK. Representamos uma visão da lateral, frente e topo.

Capítulo 7

Conclusões e Perspectivas

Nesta tese, propomos uma estrutura de visualização capaz de eliminar dados gráficos desnecessários para a visualização de uma cena 3D a fim de melhorar o desempenho de visualização.

Além de executarmos análise de eficiência dessa estrutura, propomos uma comparação com outros métodos de detecção de visibilidade e três aplicações gráficas para testá-la mais profundamente: visualização com percepção 3D, visualização de dados médicos e visualização em cavernas virtuais.

7.1 Contribuições da Tese

Temos como principal contribuição desta tese a estrutura de visualização capaz de representar cenas 3D complexas com redução significativa do desperdício de processamento de dados desnecessários para visualizá-la. Esta estrutura é capaz de detectar primitivas escondidas a partir de qualquer ponto de vista da cena. São feitas três operações que verificam se uma primitiva é dita escondida: verificar se elas se encontram dentro do volume de visualização, verificar se primitivas estão voltadas para a direção da câmera e, identificar se primitivas ocluem outras primitivas. As duas primeiras operações são bastante simples de implementar e, no caso de detecção de oclusão, determinamos sua ocorrência de duas maneiras: oclusão bloco interno e oclusão entre blocos adjacentes.

O uso de funções algébricas (provenientes da estrutura J_1^a) permitem o acesso rápido dos blocos e das faces destes blocos marcados a serem utilizados, além permitir o acesso às suas adjacências. Essa é outra contribuição, que auxilia o processamento de visualização durante a criação da lista de primitivas. É devido a essas funções que podemos navegar uma cena sem que a operação da criação da lista de primitivas afete o desempenho da visualização. Em outras palavras, podemos atualizar uma lista de primitivas rapidamente sempre que a navegação da cena cause uma necessidade de atualizar essa lista.

Os resultados obtidos demonstraram a eficiência dessa estrutura, verificamos que a taxa de quadro de visualização melhora tanto para visualização de uma cena estática quanto para uma navegação dessa cena. Verificamos também, que mesmo que as operações de remoção por *Back-Face* e *View-Frustum culling* realizem uma maior porção da remoção de primitivas, a operação de detecção oclusão ainda se faz bastante necessária

em modelos bastante complexos. Além do mais, por executarmos essas duas operações durante a etapa de preprocessamento, evitamos que a etapa de visualização execute essas mesmas operações para essa primitivas já removidas. Identificamos ainda que nosso método superestima a lista de primitivas potencialmente visíveis porém, como falaremos na Seção 7.2, nossa estrutura pode ser estendida para reduzir essa superestimação a partir de operações de refinamento dos blocos. Quando comparado à outros métodos, detectamos que o nosso método é tão ou eficiente quanto ou mais eficiente para os casos testados.

Uma consequência da redução de dados gráficos gerada por essa estrutura é que esta pode ser utilizada em aplicações baseadas em Web, onde o tempo transmissão desses dados é um fator bastante crítico. A análise que fizemos, na Seção 6.4.2, demonstra o quanto o seu uso pode melhorar o seu desempenho. Vale lembrar que, no início dessas aplicações é necessário somente carregar dados visualizados pelo ponto de vista inicial e os demais dados podem ser carregados ou em paralelo à visualização, ou a medida que forem necessários.

Finalmente, podemos dizer que a estrutura não só melhora o desempenho da visualização e transmissão de dados gráficos, como também, ela é capaz de manter a qualidade visual original de seus modelos sem gerar buracos na cena.

Como dizemos anteriormente, utilizamos a estrutura em três aplicações de visualização. Uma delas é uma aplicação nova para visualizar dados cerebrais obtidos a partir exames por DKI. Os parâmetros obtidos permitiram a geração de malhas 3D detalhadas e dos traçados das fibras da masa branca capazes de identificar o comportamento da difusão da água dentro do cérebro humano e assim identificando as diversas estruturas desse cérebro.

7.2 Trabalhos Futuros

Propõe-se como trabalhos futuros as seguintes atividades:

- Implementar a função adaptativa (presente na estrutura J_1^a) a fim de refinar blocos em possíveis regiões críticas e verificar a melhora de sua eficiência quanto ao fator de superestimação do conjunto de primitivas potencialmente visíveis. Acreditamos que refinando os blocos em certas regiões pode acarretar na criação de mais faces de blocos totalmente preenchidas aumentando mais ainda a eficiência da operação de verificação de oclusão entre blocos adjacentes;
- Utilizar objetos dinâmicos em cenas 3D e implementar um algoritmo eficiente capaz de identificar oclusão local nos blocos onde o objeto se encontra. Até o momento, identificamos essa abordagem como sendo a melhor solução para poder representar objetos dinâmicos;
- Utilizar a estrutura adicionando uma quarta dimensão para permitir a representação de animação. Essa é mais uma das razões de utilizarmos a estrutura J_1^a , já que esta é capaz de representar dimensões maiores que a terceira;
- Assim como acontece em muitos métodos baseados em região, nosso método gera uma quantidade significativa de dados extra devido a indexação de primitivas em

blocos. Propomos então, investigar métodos (como os métodos de codificação de dados 3D apresentados na Seção 2.4) de redução desses dados extras;

- Testar resultados quanto ao uso da estrutura para transmissão de dados que compõem ambientes virtuais 3D. Na literatura, a redução de dados gráficos teve maior foco na abordagem de codificação desses dados, até o momento, não conhecemos nenhum trabalho que utilize a nossa abordagem. Nesse caso, esse teste pode ser realizado em aplicações de museu virtual como a do projeto GTMV do laboratório NatalNet;
- Unir em uma só visualização as estruturas geradas pelo parâmetro de curtose média e os traçados da fibra de massa branca para gerar uma representação mais detalhada. E analisar o quanto essa representação auxilia na interpretação da informação visualizada, para isso, iremos propor uma continuidade na colaboração com os pesquisadores da Universidade de Maryland de College Park e de Baltimore.

Referências Bibliográficas

Airey, John M., John H. Rohlf & Frederick P. Brooks, Jr. (1990), 'Towards image realism with interactive update rates in complex virtual building environments', *SIGGRAPH Comput. Graph.* **24**(2), 41–50.

Alliez, Pierre & Mathieu Desbrun (2001), 'Valence-driven connectivity encoding for 3d meshes', *Computer Graphics Forum* **20**(3), 480–489.

Antani, Lakulish, Anish Chandak, Micah Taylor & Dinesh Manocha (2010), Fast geometric sound propagation with finite edge diffraction, Relatório técnico, Technical Report TR10-011, University of North Carolina at Chapel Hill, 2010. http://gamma. cs. unc. edu/BTM/FromRegion. pdf.

Assaf, Yaniv & Ofer Pasternak (2008), 'Diffusion tensor imaging (dti)-based white matter mapping in brain research : A review', *Journal of molecular neuroscience* **34**, 51–61.

Assarsson, Ulf & Tomas Möller (2000), 'Optimized view frustum culling algorithms for bounding boxes', *Journal of Graphics Tools* **5**, 9–22.

Bajaj, C., V. Pascucci & G. Zhuang (1998), Compression and coding of large cad models, Relatório técnico, University of Texas.

Bajaj, Chandrajit L, Valerio Pascucci & Guozhong Zhuang (1999), Single resolution compression of arbitrary triangular meshes with properties, *em* 'Proceedings of the Conference on Data Compression', pp. 247–256.

Bala, Kavita, Julie Dorsey & Seth Teller (1999*a*), Interactive ray-traced scene editing using ray segment trees, *em* 'Proceedings of the 10th Eurographics conference on Rendering', EGWR'99, Eurographics Association, pp. 31–44.

Bala, Kavita, Julie Dorsey & Seth Teller (1999*b*), 'Radiance interpolants for accelerated bounded-error ray tracing', *ACM Trans. Graph.* **18**(3), 213–256.

Bartz, D., J. Klosowski & D. Staneker (1999), 'Opengl-assisted occlusion culling for large polygonal models', *Computers & Graphics* **23**(5), 667 – 679.

Basser, Peter J., James Mattiello & Denis LeBihan (1994*a*), 'Estimation of the effective self-diffusion tensor from the nmr spin echo', *Journal of Magnetic Resonance* **103**, 247–254.

Basser, Peter J., James Mattiello & Denis LeBihan (1994*b*), 'Mr diffusion tensor spectroscopy and imaging', *Biophysical Journal* **66**, 259–267.

Beaulieu, Christian & Peter S. Allen (1994), 'Water diffusion in the giant axon of the squid: Implications for diffusio n-weighted mri of the nervous system', *Magnetic Resonance in Medicine* **32**, 579–583.

Bernardini, Fausto, James T. Klosowski & Jihad El-Sana (2000), 'Directional discretized occluders for accelerated occlusion culling', *Computer Graphics Forum* **19**(3), 507–516.

Bista, Sujal, Ícaro L. L. da Cunha & Amitabh Varshney (2013), 'Kinetic depth images: Flexible generation of depth perception', *Transaction on Graphics - artigo submetido* .

Bittner, J., V. Havran & P. Slavik (1998), 'Hierarchical visibility culling with occlusion trees', *Computer Graphics International Conference* **0**, 207–219.

Bittner, Jiří, Oliver Mattausch, Peter Wonka, Vlastimil Havran & Michael Wimmer (2009), Adaptive global visibility sampling, *em* 'ACM SIGGRAPH 2009 Papers', SIGGRAPH '09, pp. 94:1–94:10.

Bottasso, C. L., O. Klass & M. S. Shephard (1998), Data structures and mesh modication tools for unstructured multigrid adaptive techniques, *em* 'Engineering with Computers', Vol. 14, Springer London, pp. 234–247.

Ícaro L. L. da Cunha (2009), Estrutura de dados mate face e aplicações em geração e movimento de malhas, Dissertação de mestrado, Univesidade de São Paulo - USP-São Carlos, São Carlos, SP.

Castelo, A. (1992), Aproximação adaptativa de variedades implícias com aplicações na modelagem implícita e em equações algébrico-diferenciais, Tese de doutorado, Pontíficia Universidade Católica do Rio de Janeiro - PUC-RIO.

Castelo, A., L. G. Nonato, M. Siqueira, R. Minghim & G. Tavares (2006), 'The j1a triangulation: An adaptive triangulation in any dimension', *Computer & Graphics* **30**(5), 737–753.

Catmull, Edwin Earl (1974), A Subdivision Algorithm for Computer Display of Curved Surfaces., Tese de doutorado, The University of Utah.

Chandak, Anish, Lakulish Antani, Micah Taylor & Dinesh Manocha (2009), Fastv: From-point visibility culling on complex models, *em* 'Proceedings of the Twentieth Eurographics Conference on Rendering', EGSR'09, pp. 1237–1246.

Cheung, MM, ES Hui, KC Chan, JA Helpern, L Qi & EX Wu (2009), 'Does diffusion kurtosis imaging lead to better neural tissue characterization? a rodent brain maturation study', *Neuroimage* **45**, 386–392.

Chow, Mike M. (1997), Optimized geometry compression for real-time rendering, *em* 'Proceedings of the 8th Conference on Visualization '97', VIS '97, pp. 347–354.

Ciccarelli, Olga, Marco Catani, Heidi Johansen-Berg, Chris Clark & Alan Thompson (2008), 'Diffusion based tractography in neurological disorders: concepts, applications, and future developments', *The Lancet Neurology* **7**, 515–527.

Clark, James H. (1976), 'Hierarchical geometric models for visible surface algorithms', *Commun. ACM* **19**(10), 547–554.

Cohen-or, Daniel & Amit Shaked (1995), 'Visibility and dead-zones in digital terrain maps', *Computer Graphics Forum* **14**, 171–180.

Cohen-Or, Daniel, Eran Rich, Uri Lerner & Victor Shenkar (1996), 'A Real-Time Photo-Realistic Visual Flythrough', *IEEE Transactions on Visualization and Computer Graphics* **2**, 255–265.

Cohen-Or, Daniel, Gadi Fibich, Dan Halperin & Eyal Zadicario (1998), 'Conservative visibility and strong occlusion for viewspace partitioning of densely occluded scenes', *Computer Graphics Forum* **17**(3), 243–253.

Cohen-Or, Daniel, Yiorgos L. Chrysanthou, Cláudio T. Silva & Frédo Durand (2003), 'A survey of visibility for walkthrough applications', *IEEE Transactions On Visualization and Computer* **9**(3).

Coorg, Satyan & Seth Teller (1997), Real-time occlusion culling for models with large occluders, *em* 'Proceedings of the 1997 Symposium on Interactive 3D Graphics', I3D '97, pp. 83–90.

da Cunha, I. L. L., A. De Lacassa, V. Polizelli-Junior, J. P. Gois, L. G. Nonato & A. C. Filho (2008), Adaptive algebraic mesh generation from implicit functions, *em* 'Iberian Latin American Congress On Computational Methods in Engineering'.

Dantas, Rummenigge R., Aquiles M. F. Burlamaqui, Samuel O. Azevedo, Julio C. P. Melo, Anderson A. Souza, Luiz M. G. Gonçalves, Claudio A. Schneider, Josivan Xavier & Lucas Farias (2009), Gtmv: Virtual museum authoring systems, *em* 'Proceedings of the 2009 IEEE International Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems', VECIMS'09, pp. 129–133.

Deering, Michael (1995), Geometry compression, *em* 'Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques', SIGGRAPH '95, ACM, New York, NY, USA, pp. 13–20.

Durand, Frédo, George Drettakis, Joëlle Thollot & Claude Puech (2000), Conservative visibility preprocessing using extended projections, *em* 'Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques', SIGGRAPH '00, pp. 239–248.

Falangola, MF, JH Jensen, JS Babb, C Hu, FX Castellanos, A Di Martino, SH Ferris & JA Helpern (2008), 'Age-related non-gaussian diffusion patterns in the prefrontal brain', *J Magn Reson Imaging* **28**, 1345–1350.

Foley, James D., Andries van Dam, Steven K. Feiner & John F. Hughes (1990), *Computer graphics: principles and practice (2nd ed.)*, Addison-Wesley Longman Publishing Co., Inc.

Garlick, B., D. D. Baum & J. Winget (1990), 'Interactive viewing of large geometric data bases using multiprocessor graphics workstations'.

Gois, J. P., V. Polizelli-Junior, T. Etiene, E. Tejada, A. Castelo, T. Ertl & L. G. Nonato (2007), Robust and adaptive surface reconstruction using partition of unity implicits, *em* 'SIBGRAPI '07: Proceedings of the XX Brazilian Symposium on Computer Graphics and Image Processing', IEEE Computer Society, pp. 95–104.

Gomes, Jonas & Luiz Velho (1990), *Computação Gráfica Vol.1*, Série de Computação e Matemática. IMPA.

Gonçalves, L. M. G. (1996), Modelagem de terreno com triangulação cfk adaptativa, *em* 'Anais do IX Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens', pp. 337–338.

Gonçalves, L. M. G. (1997), Novos resultados em triangulações (triangulação gulosa), *em* 'Anais do X Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens'.

Greene, N. (1999), Occlusion culling with optimized hierachical z-buffering, *em* 'In ACM SIGGRAPH Visual Proceedings 1999'.

Greene, N. (2001*a*), 'Occlusion culling with optimized hierachical z-buffering'.

Greene, N. (2001*b*), 'A quality knob for non-conservative culling with hierarchical z-buffering'.

Greene, Ned & Michael Kass (1994), Error-bounded antialiased rendering of complex environments, *em* 'Proceedings of the 21st annual conference on Computer graphics and interactive techniques', SIGGRAPH '94, ACM, pp. 59–66.

Greene, Ned, Michael Kass & Gavin Miller (1993), Hierarchical z-buffer visibility, *em* 'Proceedings of the 20th annual conference on Computer graphics and interactive techniques', SIGGRAPH '93, ACM, New York, NY, USA, pp. 231–238.

Gumhold, S. (1999), Improved cut-border machine for triangle mesh compression, *em* 'Erlangen Workshop 99 on Vision, Modeling and Visualization, IEEE Signal Processing Society'.

Gumhold, Stefan & Wolfgang Strasser (1998), Real time compression of triangle mesh connectivity, *em* 'Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques', SIGGRAPH '98, pp. 133–140.

Horsfield, Mark A. & Derek K. Jones (2002), 'Applications of diffusion-weighted and diffusion tensor mri to white matter diseases - a review', *NMR in Biomedicine* **15**, 570–577.

Hudson, T., D. Manocha, J. Cohen, M. Lin, K. Hoff, & H. Zhang (1997), Accelerated occlusion culling using shadow frustra, *em* 'Proc. 13th Annu. ACM Sympos. Comput. Geom.', ACM, pp. 1–10.

Inglese, M. & M. Bester (2010), 'Diffusion imaging in multiple sclerosis: research and clinical implications', *NMR in Biomedicine* **23**, 865–872.

Jensen, J.H., J.A. Helpern, A. Ramani, H. Lu & K. Kaczynski (2005), 'Diffusional kurtosis imaging: The quantification of non-gaussian water diffusion by means of magnetic resonance imaging', *Magnetic Resonance in Medicine* **53**(6), 1432–1440.

Jones, C. B. (1971), 'A new approach to the 'hidden line' problem', *The Computer Journal* **14**(3), 232–237.

King, Davis & Jarek Rossignac (1999), Guaranteed 3.67v bit encoding of planar triangle graphs, *em* '11th Canadian Conference on Computational Geometry', pp. 146–149.

Klosowski, J.T. & C.T. Silva (2000), 'The prioritized-layered projection algorithm for visible set estimation', *Visualization and Computer Graphics, IEEE Transactions on* **6**(2), 108–123.

Klosowski, J.T. & C.T. Silva (2001), 'Efficient conservative visibility culling using the prioritized-layered projection algorithm', *Visualization and Computer Graphics, IEEE Transactions on* **7**(4), 365–379.

Koltun, Vladlen, Yiorgos Chrysanthou & Daniel Cohen-Or (2000), Virtual occluders: An efficient intermediate pvs representation, *em* B.Péroche & H.Rushmeier, eds., 'Rendering Techniques 2000', Eurographics, Springer Vienna, pp. 59–70.

Koltun, Vladlen, Yiorgos Chrysanthou & Daniel Cohen-Or (2001), Hardware-accelerated from-region visibility using a dual ray space, *em* 'Proceedings of the 12th Eurographics Workshop on Rendering Techniques', pp. 205–216.

Kubicki, Marek, Robert McCarley & C-F. Westin (2007), 'A review of diffusion tensor imaging studies in schizophrenia', *Journal of Psychiatric Research* **41**, 15–30.

Kumar, Subodh, Dinesh Manocha, William Garrett & Ming Lin (1996), Hierarchical back-face computation, *em* 'Proceedings of the eurographics workshop on Rendering techniques '96', Springer-Verlag, London, UK, pp. 235–244.

Kwok, W., K. Haghighi & E. Kang (1995), An efcient data structure for the advancing-front triangular mesh generation technique, *em* 'Communications in Numerical Methods in Engineering', Vol. 11, pp. 465–473.

Lange, N, Dubray MB, Lee JE, Froimowitz MP, Froehlich A, Adluru N, Wright B, Ravichandran C, Fletcher PT, Bigler ED, Alexander AL & Lainhart JE (2010), 'Atypical diffusion tensor hemispheric asymmetry in autism', *Autism Res* **3**(6), 350–358.

Le Bihan, D (1991), 'Molecular diffusion nuclear magnetic resonance imaging', *Magn Reson Q* **7**, 1–30.

Le Bihan, D, E Breton, D Lallemand, P Grenier, E Cabanis & M Laval-Jeantet (1986), 'Mr imaging of intravoxel incoherent motions: application to diffusion and perfusion in neurologic disorders', *Radiology* **161**, 401–407.

Le Bihan, D. & PJ Basser (1995), *Molecular diffusion and nuclear magnetic resonance*, Raven Press, capítulo Diffusion and perfusion magnetic resonance imaging: Applications to functional MRIch, pp. 5–17.

Luebke, David & Chris Georges (1995), Portals and mirrors: simple, fast evaluation of potentially visible sets, *em* 'Proceedings of the 1995 symposium on Interactive 3D graphics', I3D '95, ACM, pp. 105–106.

Maller, Jerome J., Richard H.S. Thomson, Philip M. Lewis, Stephen E. Rose, Kerstin Pannek & Paul B. Fitzgerald (2010), 'Traumatic brain injury, major depression, and diffusion tensor imaging: Making connections', *Brain Research Reviews* **64**, 213–240.

Meagher, D. (1982), 'Efficient synthetic image generation of arbitrary 3-d objects', *IEEE Computer Society Conference on Pattern Recognition and Image Processing* pp. 473–478.

Mäntylä, M. (1988), *An Introduction to Solid Modeling*, Computer Science Press.

Neves, M. R. & R. M. Persiano (1997), Visualizing scalar fields represented by adaptive square triangulations, *em* 'Proceedings of the X Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI)'.

Parker, Steven, William Martin, Peter pike J. Sloan, Peter Shirley, Brian Smits & Charles Hansen (1999), Interactive ray tracing, *em* 'In Symposium on interactive 3D graphics', pp. 119–126.

Peng, Jingliang, Chang-Su Kim & C. C. Jay Kuo (2005), 'Technologies for 3d mesh compression: A survey', *J. Vis. Comun. Image Represent.* **16**(6), 688–733.

Persiano, R. M., J. Luiz, D. Comba & V. Barbalho (1993), An adaptive triangulation refinement scheme and construction, *em* 'Proceedings of the VI Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI)', pp. 259–266.

Pierpaoli, C, P Jezzard, PJ Basser, A Barnett & G Di Chiro (1996), 'Diffusion tensor mr imaging of the human brain', *Radiology* **201**, 637.

Rossignac, Jarek (1999), 'Edgebreaker: Connectivity compression for triangle meshes', *IEEE Transactions on Visualization and Computer Graphics* **5**(1), 47–61.

Schaufler, Gernot, Julie Dorsey, Xavier Decoret & François X. Sillion (2000), Conservative volumetric visibility with occluder fusion, *em* 'Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques', SIGGRAPH '00, pp. 229–238.

Slater, M. & Y. Chrysanthou (1997), 'View volume culling using a probabilistic cashing scheme', *ACM Virtual Reality Software and Technology VRST'97* pp. 71–78.

Taubin, Gabriel & Jarek Rossignac (1998), 'Geometric compression through topological surgery', *ACM Trans. Graph.* **17**(2), 84–115.

Teller, Seth J (1992), Visibility computations in densely occluded polyhedral environments, Relatório técnico, Berkeley, CA, USA.

Teller, Seth J. & Carlo H. Séquin (1991), 'Visibility preprocessing for interactive walkthroughs', *SIGGRAPH Comput. Graph.* **25**(4), 61–70.

Tian, Fenglin, Wei Hua, Zilong Dong & Hujun Bao (2010), 'Adaptive voxels: Interactive rendering of massive 3d models', *Vis. Comput.* **26**(6-8), 409–419.

Touma, C. & C. Gotsman (1998), Triangle mesh compression, *em* 'Proceedings of Graphics Interface', pp. 26–34.

Tuch, DS (2002), 'High angular resolution diffusion imaging reveals intravoxel white matter fiber heterogeneity', *Magnetic Resonance in Medicine* **48**, 577–582.

Turán, Gyorgy (1984), 'On the succinct representation of graphs', *Discrete Applied Mathematics* **8**(3), 289 – 294.

Westlye, Lars T, Kristine B Walhovd, Anders M Dale, Atle Bjornerud, Paulina Due-Tonnessen, Andreas Engvig, Hakon Grydeland, Christian K Tamnes, Ylva Ostby & Anders M Fjell (2010), 'Life-span changes of the human brain white matter: Diffusion tensor imaging (dti) and volumetry', *Cerebral Cortex* **20**(9), 2055–2068.

Wiegell, Mette R. and; Larsson, Henrik B. W. & J. Wedeen Van (2000), 'Fiber crossing in human brain depicted with diffusion tensor mr imaging', *Radiology* **217**, 897–903.

Wonka, Peter & Dieter Schmalstieg (1999), 'Occluder shadows for fast walkthroughs of urban environments', *Computer Graphics Forum* **18**(3), 51–60.

Wonka, Peter, Michael Wimmer & Dieter Schmalstieg (2000), Visibility preprocessing with occluder fusion for urban walkthroughs, *em* 'Proceedings of the Eurographics Workshop on Rendering Techniques 2000', pp. 71–82.

Zhang, Hansong (1998), Effective Occlusion Culling for the Interactive Display of Arbitrary Models, Tese de doutorado, Department of Computer Science, UNC-Chapel Hill.

Zhuo, J., S. Xu, J.L. Proctor, R.J. Mullins, J.Z. Simon, G. Fiskum & R.P. Gullapalli (2012), 'Diffusion kurtosis as an in vivo imaging marker for reactive astrogliosis in traumatic brain injury', *Neuroimage* **59**(1), 467–477.