Pontifícia Universidade Católica do Rio de Janeiro

Departamento de Informática

DISSERTAÇÃO DE MESTRADO

Multi-resolução para a Visualização de Reservatórios Naturais de Petróleo

Aluno: Antonio Carlos Pereira de Azambuja

azambuja@tecgraf.puc-rio.br

Orientadores: Prof. Marcelo Gattass gattass@ tecgraf.puc-rio.br

Prof. Waldemar Celes celes@ tecgraf.puc-rio.br

Rio de Janeiro, Maio de 2002.

Antonio Carlos Pereira de Azambuja

Multi-resolução para a Visualização de Reservatórios Naturais de Petróleo

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática do Departamento de Informática da PUC-Rio.

Orientadores: Marcelo Gattass e Waldemar Celes Filho

Rio de Janeiro
Maio de 2002

Antonio Carlos Pereira de Azambuja

Graduou-se em Informática na UFSM (Universidade Federal de Santa Maria - RS) em 1998. Desde então, vem participando de cursos na área de Computação de Gráfica, no IMPA (Instituto de Matemática Pura e Aplicada) e na PUC-Rio. Desde 1999, trabalha em desenvolvimento de sistemas no TeCGraf (Laboratório de Computação Gráfica da PUC-Rio), ocupando inicialmente o cargo de pesquisador júnior e, posteriormente, de analista. Durante esse período, participou do desenvolvimento dos sistemas SIGMA (Sistema Integrado de Gestão de Meio Ambiente) e PósSim (Pós-visualizador da Simulação de Reservatórios Naturais de Petróleo), ambos em parceria com a Petrobras (Companhia de Petróleo do Brasil).

Ficha Catalográfica

Azambuja, Antonio Carlos Pereira de

Multi-resolução para a Visualização de Reservatórios Naturais de Petróleo \ Antonio Carlos Pereira de Azambuja; orientadores: Marcelo Gattass e Waldemar Celes. Rio de Janeiro: PUC, Departamento de Informática, 2002.

vii., 137 f.: il. ; 29,7 cm

1. Dissertação (mestrado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui referências bibliográficas.

1. Informática – Dissertações. 2. Multi-resolução. 3. Visualização Científica. 4. Reservatórios naturais de petróleo. 5. Simulação do fluxo de fluido em meios porosos. 6. Descarte. I. Gattass, Marcelo. II. Celes, Waldemar. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

Antonio Carlos Pereira de Azambuja

Multi-resolução para a Visualização de Reservatórios Naturais de Petróleo

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática do Departamento de Informática da PUC-Rio.

Orientadores: Marcelo Gattass e Waldemar Celes Filho

Rio de Janeiro
Maio de 2002

Para meus pais, Tércio e Suzara, pelo
apoio constante e para Juliana,
meu amor.

Agradecimentos

A Marcelo Gattass pela confiança depositada desde a minha chegada no Rio de Janeiro, pela importantíssima ajuda em momentos críticos e decisivos da minha entrada no Mestrado e pela valiosa orientação, principalmente durante a confecção do texto final desta dissertação.

A Waldemar Celes Filho pela orientação técnica impecável, pela supervisão constante, sutil e motivadora de todo o processo de pesquisa e desenvolvimento deste trabalho, sem as quais a qualidade dos resultados estaria comprometida.

A Paulo Cezar Carvalho e Leo Schneider por terem indicado a mim os possíveis caminhos a serem seguidos durante essa etapa de Mestrado, participando, assim, decisivamente desse processo.

A Banca examinadora desta dissertação pela correções, dicas e sugestões que, muitas vezes, foram feitas na etapa de desenvolvimento do trabalho.

A Carolina Alfaro pelas dicas de linguagem e pela paciência e eficiência dispensadas na correção ortográfica e sintática do texto.

A todos os meus professores de Graduação e Mestrado e meus colegas do TeCGraf pelo inestimável conhecimento passado durante os últimos anos.

A CAPES, Petrobras e PUC elo estímulo e apoio financeiro oferecidos, direta ou indiretamente, sem os quais este trabalho não se realizaria.

Aos meus pais, Tércio e Suzara, pelo amor infinito, carinho acolhedor, atenção tranqüilizante e apoio incondicional a mim direcionados durante toda a vida.

A Juliana, meu amor, pela cumplicidade, carinho, afeto, amizade, segurança, pelos momentos de diversão, pelos sorrisos e choros, pela devoção mútua, pela paz interior que o nosso convívio possibilita e por tudo que isso representa.

A minhas irmãs, Melissa e Carolina, pela cumplicidade natural que temos em tudo que fazemos e pelo prazer que é poder constatar isso.

A minha segunda família, Telmo, Ida, Bianca, Geórgia, Christian, Eduardo, Letícia, Luiz Rodolfo, João Pedro, José Luiz e Antonio Pedro, pelos momentos maravilhosos compartilhados, pelo amor, pelo carinho e pelo apoio infindável.

A meus amigos de estudo e trabalho pelos felizes e valiosos momentos convividos. A meus distantes amigos “de sempre” pela torcida que fizeram para o meu sucesso e pelo companheirismo fraterno.

Ao universo e suas possibilidades.

Resumo

Atualmente, as malhas de simulação do fluxo em reservatórios naturais de petróleo (**RNPs**) são modelos compostos por centenas de milhares de células hexaédricas, cada uma podendo ser decomposta em 12 triângulos, de modo que a visualização interativa dessas malhas, através das estações gráficas atuais, ainda não é factível. À medida que os computadores e as placas gráficas aumentam sua capacidade de processamento, as malhas de simulação também crescem. A solução para esse tipo de problema passa, normalmente, por técnicas de aceleração, dentre as quais está a multi-resolução (**MR**). Ocorre, entretanto, que os modelos de multi-resolução atualmente conhecidos não são aplicáveis às malhas de simulação de RNPs, devido aos requisitos específicos da área, tais como a preservação do modelo de células hexaédricas e a descontinuidade entre células. Na realidade, as técnicas de multi-resolução tendem a enfocar a Visualização Realista, enquanto o problema de RNPs é de Visualização Científica, para a qual ainda não existem soluções genéricas.

Esta dissertação propõe um modelo de MR específico para o problema de visualização das malhas de simulação em RNPs, no qual a partição descontínua do espaço, a semântica baseada em células hexaédricas e as características de visualização do problema são pontos considerados. O modelo proposto permite uma construção eficiente da estrutura de MR, a partir da qual, em tempo real, são extraídas malhas adaptativas dependentes: (a) do erro geométrico da aproximação, (b) da câmera e (c) do número desejado de polígonos na malha. Além disso, o modelo permite a utilização conjunta de outra técnica de aceleração, o descarte, possibilitando o descarte hierárquico de regiões da malha que estão fora do volume de visão. O modelo proposto foi implementado em um sistema que permitiu uma extensa bateria de testes, cujos resultados permitiram traçar algumas conclusões e recomendações.

Palavras-chave

Técnicas de aceleração; multi-resolução; descarte; malhas de simulação; simulação; reservatórios naturais de petróleo; visualização científica.

Abstract

Current flow-simulation meshes of natural oil reservoirs (NOR) are composed of hundreds of thousands of hexahedral cells. The visualization of the geometry of these cells superimposed with color attributes to represent properties and flow results requires the rendering of an unstructured mesh of millions of triangles. Current graphics hardware does not allow for an interactive visualization of such meshes. As computers and graphics boards increase their processing capacity, simulation meshes also grow and the solution to the rendering problem usually includes acceleration techniques, one of which is multi-resolution (MR). However, currently known MR models are not applicable to NOR simulation meshes due to this field's specific requirements, such as the preservation of the hexahedral-cell model and discontinuities among cells. In fact, MR techniques tend to focus on Realistic Visualization, while the NOR problem is one of Scientific Visualization, for which generic solutions still do not exist.

The present work proposes a specific MR model for the visualization problem concerning NOR simulation meshes, in which discontinuous space partition, hexahedral-cell-based semantics and the problem's visualization characteristics are taken into account. The proposed model allows an efficient construction of a MR structure, from which, in real time, adaptive meshes can be extracted that depend on: (a) the geometric error approximation, (b) the view, and (c) the polygon budget. This model can also be used combined with another acceleration technique, frustum culling, which allows for the hierarchical elimination of regions in the mesh that are out of the view volume. The proposed model was implemented in a system on which extensive testing was performed, providing results that allowed us to draw some conclusions and recommendations.

Keywords

Acceleration techniques, multi-resolution, frustum culling, simulation meshes, simulation, natural oil reservoir, scientific visualization.

Sumário

Lista de figuras

Lista de Tabelas

Lista de Gráficos

Lista de Conjuntos de Gráficos

1 Introdução

As malhas de simulação do fluxo de fluidos em reservatórios naturais de petróleo (**RNPs**) são modelos compostos por até centenas de milhares de células, sendo que, com o constante desenvolvimento dos métodos de simulação e aquisição de dados (tais como sísmica e prospecção), há um aumento sistemático dessa quantidade ao longo do tempo.

A visualização das malhas de simulação requer taxas de interação que não são conseguidas com os atuais computadores comumente utilizados (PCs e estações de trabalho de uso geral). Para resolver este problema, podem-se utilizar técnicas de aceleração, dentre as quais destacam-se (a) o uso de um esquema de representação em diferentes **níveis de detalhe** no qual a malha original é substituída por versões menos refinadas e (b) o descarte de regiões da malha que estão fora do volume de visão (*frustum culling*).

A primeira técnica caracteriza os modelos de **multi-resolução**, os quais, sob o ponto de vista de uma técnica de aceleração, devem ser capazes de manter uma estrutura compacta de representação através da qual simplificam-se regiões cujas projeções na tela ocupam áreas pequenas. Diversos algoritmos de multi-resolução foram propostos nos últimos anos para tratar tipos distintos de objetos representados pelo paradigma poligonal. A literatura é vasta e vem se tornando estruturada devido à formalização dos conceitos utilizados e à criação de *frameworks* bastante expressivos [Flo97, Pup97].

Obtiveram-se resultados substanciais para tratar superfícies e volumes de topologia *manifold*[1] imersos no espaço tridimensional. Garland [Gar99] salienta que há uma tendência de considerar complexos simpliciais aos quais é aplicada uma série de transformações locais (Figura 1.1), alterando-os gradualmente por simplificação ou refinamento. Os trabalhos de Hoppe [Hop97, Hop96] são exemplos expressivos aplicados ao caso de superfícies e os de Cignoni [Cig98, Cig97] ao caso volumétrico.

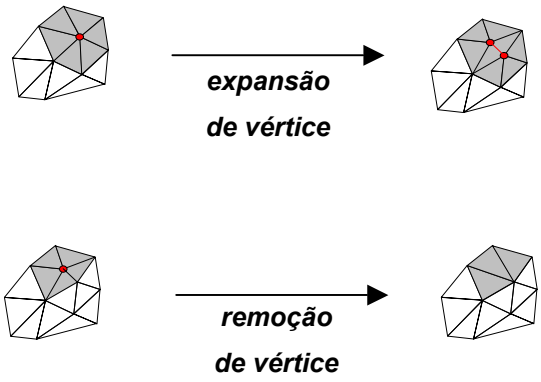
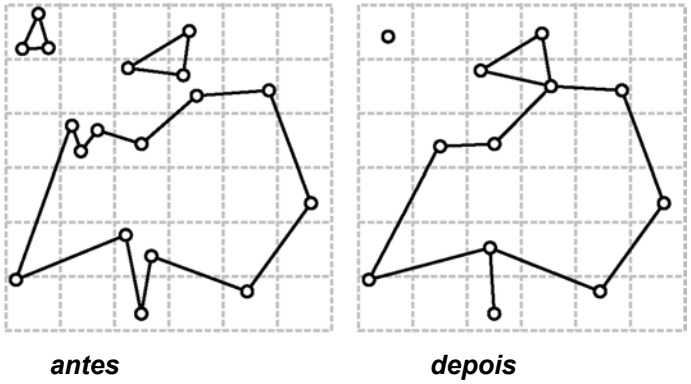


Figura 1.1- Transformações locais tipicamente aplicadas a superfícies *2-manifold*.

Para malhas de topologia *non-manifold*, os algoritmos de multi-resolução adotam diferentes abordagens, que variam desde a superposição de uma grade regular, fazendo com que vértices pertencentes à mesma célula na grade sejam colapsados (*vertex clustering*) [Rossignac93, Low97, Sch95] (Figura 1.2-a), até a aplicação de técnicas voltadas a *manifold* que apenas conectam vários componentes disjuntos gerando superfícies *non-manifold* [Gar99] (Figura 1.2-b). Poucos algoritmos tratam essa topologia de forma explícita, sendo que um dos trabalhos mais recentes nesse sentido é o de Hoppe e Popovíc [Hop98], que considera também o uso de células de diferentes dimensões em uma mesma malha.

---

[1] *Uma superfície S é* manifold *se* $\forall p \in S$, *a vizinhança aberta de p é homeomorfa ao círculo unitário aberto ou ao semi-círculo unitário, resultado da interseção entre o círculo unitário aberto e o semi-plano fechado com x positivo. A definição de um volume* manifold *é análoga à de superfície, entretanto utiliza-se a bola aberta e a semi-bola aberta [Math].*



(a)

(b)

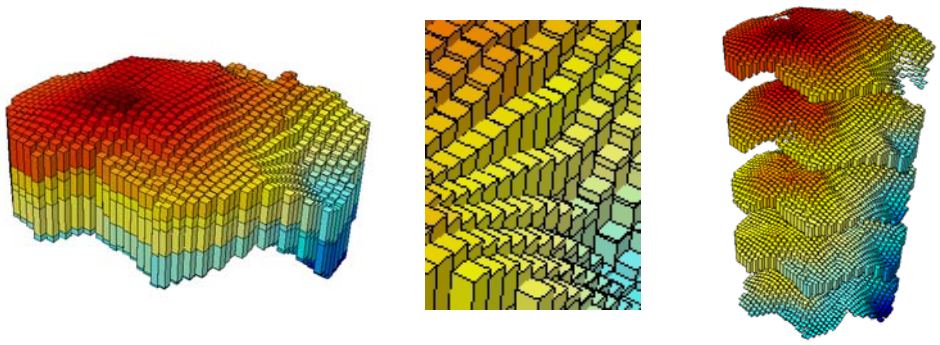
Figura 1.2 - Em (a), o algoritmo de *vertex clustering* (imagem retirada de [Gar99]). Em (b), um exemplo de operação que permite a geração de superfícies *non-manifold*.

A visualização das malhas de simulação em reservatórios se caracteriza pela visualização da fronteira de células hexaédricas dotadas de uma relação de vizinhança definida explicitamente: cada célula é identificada por uma tripla **<*i*, *j*, *k*>** e é vizinha das células **<*i*+1, *j*, *k*>**, **<*i*-1, *j*, *k*>**, **<*i*, *j*+1, *k*>**, **<*i*, *j*-1, *k*>**, **<*i*, *j*, *k*+1>** e **<*i*, *j*, *k*-1>**. Tipicamente, células vizinhas possuem descontinuidades de forma que o arranjo das células forme uma malha **não-conforme**[2] e o conjunto de suas fronteiras, globalmente, tenha topologia *non-manifold* (Figura 1.3-a).

É interessante para a visualização que as descontinuidades entre células sejam fielmente representadas, principalmente quando uma região pequena do reservatório está em foco (Figura 1.3-b). Além disso, há a necessidade de visualização de entidades especiais formadas por agrupamentos de células, os quais são identificados através das relações de vizinhança entre células, cujo exemplo típico é o agrupamento

[2] *Uma malha é dita* conforme *se faces adjacentes no espaço compartilham exatamente arestas e vértices em bordas comuns [Velho00].*

por camadas de sedimentação que, para serem visualizadas, são separadas explicitamente no espaço através de translações (Figura 1.3-c).



(a) (b) (c)

Figura 1.3 - Visualização de RNPs. Em (a), todo o reservatório. Em (b), uma região em foco. Em (c), separação explícita entre camadas.

Assim, devido à topologia da malha e à necessidade de visualização de suas descontinuidades, os algoritmos de multi-resolução baseados em *manifold* não são diretamente aplicáveis a reservatórios. Por outro lado, os algoritmos baseados em transformações como colapso de vértices (Figura 1.2-b) tratariam a malha de simulação como uma malha qualquer de polígonos, perdendo a semântica da célula hexaédrica e de suas relações de vizinhança. Neste sentido, estruturas clássicas como *octrees* e *quadtrees* são opções possíveis, mas não geram aproximações bem adaptadas a geometrias complicadas [Pup97], como é o caso das malhas de simulação em reservatórios.

Percebe-se, portanto, a ausência de algoritmos de multi-resolução adequados às características peculiares da visualização das malhas de simulação do fluxo de fluidos em RNPs. Somando-se a isto a necessidade latente de aumento das taxas de interação obtidas atualmente, tem-se um contexto que justifica o estudo e o projeto de um algoritmo de multi-resolução específico para a visualização de reservatórios.

1.1 Objetivos

O objetivo deste trabalho é propor e analisar um modelo de multi-resolução que responda aos requisitos específicos da visualização das malhas de simulação em reservatórios. O modelo deve levar em consideração as características peculiares dessas malhas, observando a partição descontínua do espaço, a semântica baseada na célula hexaédrica e a relação de vizinhança entre células e possibilitando a extração de malhas adaptativas em tempo real controlada pelo erro geométrico da aproximação, pela posição da câmera e pelo número desejado de polígonos contidos na malha.

1.2 Organização do Texto

Visando apresentar alguns quesitos pertinentes ao assunto proposto e outros necessários a uma boa compreensão deste trabalho, o texto é organizado da seguinte forma: o Capítulo 2 apresenta um estudo sobre modelos de multi-resolução, definindo o problema de multi-resolução e caracterizando dois modelos bastante significativos. Os conceitos e nomenclaturas nele definidos serão utilizados posteriormente.

O Capítulo 3 analisa os reservatórios naturais de petróleo, objetivando oferecer um entendimento, mesmo que superficial, a respeito do fenômeno físico relacionado aos RNPs e caracterizar as malhas de simulação em reservatórios com sua geometria e propriedades. Nesta ocasião, os objetivos a serem alcançados pelo modelo de multi-resolução proposto são definidos com precisão.

Nos três capítulos seguintes, o modelo de multi-resolução é proposto e analisado, e os resultados são apresentados. O Capítulo 4 concentra-se na definição e análise do modelo. No Capítulo 5, os algoritmos de extração de malhas adaptativas a partir do modelo proposto são explicados e analisados. E, no Capítulo 6, apresenta-se o estudo de caso de uma implementação desse modelo de multi-resolução. Tanto os resultados de desempenho quanto os de qualidade das aproximações são analisados.

Finalmente, o Capítulo 7 apresenta as conclusões obtidas.

2 Modelos de Multi-resolução

Os objetivos deste capítulo são (a) apresentar o problema de multi-resolução (**MR**) de entidades geométricas representadas por modelos poligonais/poliedrais, tais como as malhas de simulação em RNPs, e (b) caracterizar dois tipos de modelos de MR (**MMRs**): os hierárquicos e os reticulares.

Os MMRs tratados aqui são denominados por Floriani [Flo97] e Puppo [Pup97] **MMR Geométricos**, em contraposição a outras abordagens existentes para a MR como, por exemplo, a de espaço de escala [Lin94], baseada na teoria dos sinais. Além disso, o enfoque principal é dado à MR sob o ponto de vista de uma técnica de aceleração.

É necessário explicitar que este trabalho faz uso de conceitos relacionados à representação paramétrica de objetos, definida por uma função $\mathbf{o}:\mathbf{U} \subset \mathfrak{R}^n \rightarrow \mathfrak{R}^m$, onde **U** é chamado de **domínio do objeto** ou **espaço paramétrico**, e $\mathfrak{R}^m$ é chamado de **espaço geométrico do objeto**. Por exemplo, algumas superfícies imersas no $\mathfrak{R}^3$ podem ser representadas pela função $\mathbf{o}:\mathbf{U} \subset \mathfrak{R}^2 \rightarrow \mathfrak{R}^3$ (Figura 2.1). Dessa forma, os termos **domínio** e **modelo** são utilizados para representar, respectivamente, o objeto no espaço paramétrico e no espaço geométrico.

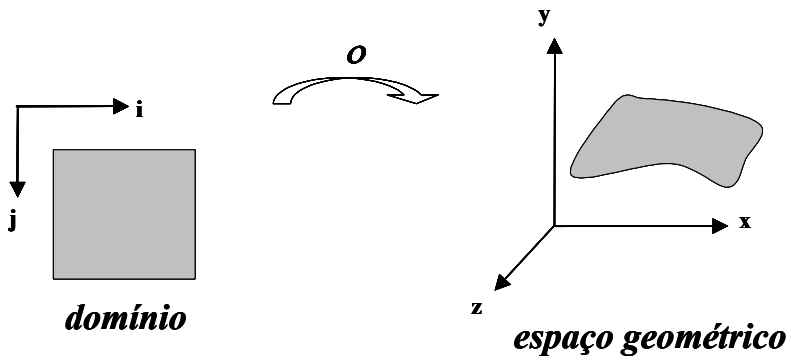


Figura 2.1 – Superfície paramétrica.

2.1 Multi-resolução

Um MMR é um mecanismo que suporta a representação e o processamento de entidades geométricas em diferentes níveis de detalhe [Pup97]. Garland [Gar99] formalizou o conceito de MMR através da família de modelos

$$\mathbf{M}:\mathbf{C}\rightarrow\mu$$

onde **C** representa os contextos de visualização (parâmetros de câmera e descrição de cena, por exemplo) e μ é o espaço não-enumerável de todos os modelos existentes que representam um mesmo objeto. Assim, $\mathbf{M}(\nu) \in \mu$ representa o modelo do objeto que é adequado ao contexto $\nu \in \mathbf{C}$.

Devido à grande afinidade entre os problemas de MR e os de simplificação poligonal[3] e com base nos resultados de Suri e Agarwal [Aga94], que provaram que o problema de simplificação poligonal baseado na norma de erro L_∞ para campos escalares é NP-Hard, estima-se que achar $\mathbf{M}(\nu)$ de forma ótima seja também um problema NP-Hard [Gar99].

Para contornar este problema, geralmente utilizam-se heurísticas que sejam capazes de selecionar um subconjunto finito $\mu' \subset \mu$, tal que μ' seja satisfatório em todos os

---

[3] Dado um espaço μ de modelos que representam um mesmo objeto e um modelo de referência $\Sigma_0 \in \mu$, o problema de simplificação poligonal pode ser assim definido: achar o modelo $\Sigma \in \mu$ com o menor número possível de células, tal que $\varepsilon(\Sigma_0, \Sigma) < \Delta$, onde ε é a função que define o erro entre dois modelos e Δ é o limite máximo permitido para o erro [Math].

contextos de visualização. Esse processo será denominado, neste trabalho, **construção do espaço discreto de modelos** e é comumente feito em pré-processamento.

A construção, a armazenagem e a codificação de μ' devem ser tais que permitam, preferencialmente, dentre outras coisas:

- uma construção eficiente do espaço discreto de modelos;

- uma representação compacta da estrutura de dados – geralmente, considera-se razoável que a quantidade de memória utilizada seja um pouco maior (em torno do dobro) do que a quantidade de memória necessária para armazenar unicamente o modelo de maior resolução [Gar99]; e

- um mecanismo eficiente de extração, em tempo de exibição, de modelos adaptativos dependentes do contexto de visualização.

A principal idéia proposta na literatura para caracterizar um MMR é definir o espaço discreto de modelos μ' a partir de um esquema de decomposição do domínio no qual a resolução de cada modelo é controlada pelo grau de refinamento da decomposição [Flo97] (Figura 2.2). Ou seja, o número de células da decomposição de cada modelo, de forma geral, diz qual é a sua resolução.

Esse esquema é apropriado para modelos dados por decomposições do espaço e, particularmente, para outros tipos de decomposições não contínuas (como as malhas de simulação em reservatórios), nas quais cada célula está associada a uma porção do domínio do objeto, formando a unidade de fragmentação da decomposição.

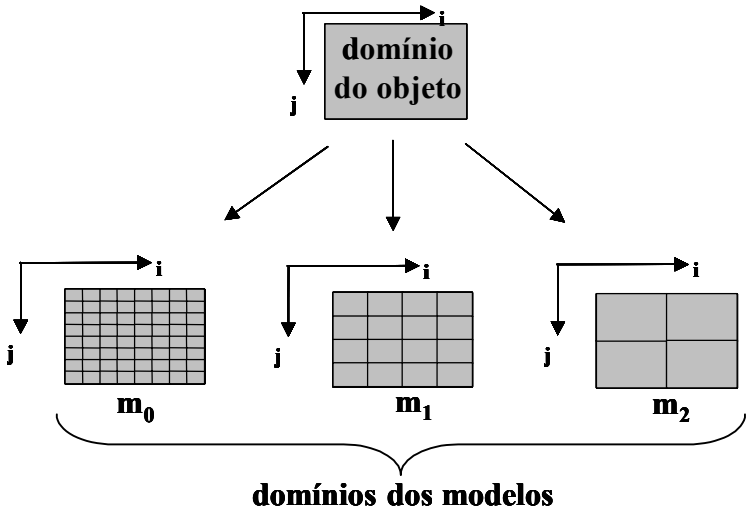


Figura 2.2 – Esquema de decomposição do domínio.

É importante para o MMR dispor de uma quantidade de fragmentos que lhe ofereça o **poder de expressão**[4] necessário. Dado o modelo inicial de um objeto, com a quantidade de fragmentos suficiente para gerar uma única representação desse objeto, é tarefa do MMR gerar novos fragmentos e estruturá-los adequadamente. Este processo corresponde exatamente à construção do espaço discreto de modelos.

Os diversos algoritmos existentes para resolver este problema são baseados, de uma forma ou de outra, em dois tipos de alterações [Pup97, Gar98]: **alterações locais**, que atuam sobre apenas uma parte do domínio (pequeno grupo de células), e **alterações globais**, que atuam sobre todo o domínio (todas, ou quase todas, as células do modelo).

A utilização extrema da idéia de alterações globais cria MMRs caracterizados pela existência de um número fixo, normalmente pequeno, de representações distintas de um mesmo objeto, sendo classificados como **discretos**. Por outro lado, o uso de alterações locais proporciona a criação de MMRs caracterizados pela existência de um número elevado, virtualmente infinito, de representações distintas, sendo classificados como **contínuos**.

---

[4] *O poder de expressão de um MMR é igual à cardinalidade de μ' [Velho00].*

Neste contexto, existem dois MMRs que abrangem a maioria dos algoritmos de MR propostos na literatura, encerrando em si importantes problemas e conceitos da área: **modelos hierárquicos** e **modelos reticulares** [5] [Flo97].

2.2 Modelos de Multi-resolução Hierárquicos

Consideremos um **modelo base** Σ_0 que seja a aproximação inicial de um objeto. Os modelos de multi-resolução hierárquicos (**MMRHs**) são construídos a partir da decomposição gerada pelo modelo base através de **regras de modificação** que definem padrões para alterar cada célula da decomposição, modificando sua conectividade e inferindo a nova geometria. Neste sentido, cada célula $c_i \in \Sigma_0$ é vista como uma entidade independente que abrange um domínio $dom(c_i)$.

As regras de modificação predizem a forma com que as células são modificas através de **padrões de modificação**. Um padrão pode ser de simplificação (cuja aplicação, na prática, só é possível em alguns casos bem definidos, como o do algoritmo *mipmapping* [Wil83]) ou o de refinamento. Sem perda de generalidade, consideremos apenas o caso do refinamento, no qual Σ_0 é o modelo menos refinado. Um padrão de refinamento é aplicado à célula c_i, substituindo-a por um novo conjunto de células que formam o modelo Σ_j, que abrange exatamente o domínio $dom(c_i)$ (Figura 2.3). Este padrão é aplicado novamente a cada célula de Σ_j e o processo é repetido recursivamente até que o domínio $dom(c_i)$ esteja representado na resolução máxima necessária para a aplicação, estando próximo o suficiente do objeto segundo a norma de erro utilizada.

---

[5] *Originalmente, Floriani [Flo97] referencia o modelo reticular através do termo modelo piramidal (pyramidal model). O termo reticular é, aqui, utilizado por motivos de clareza e preferência pessoal. Outras denominações existentes para os modelo hierárquicos e reticulares são, respectivamente, modelos representados por árvores e modelos baseados em históricos [Pup97].*

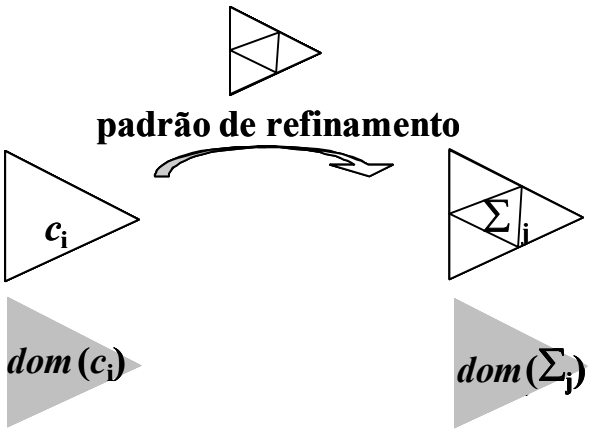


Figura 2.3 - Padrão de refinamento conhecido como triangulação quaternária.

A estrutura natural para representar o processo de refinamento é uma árvore. Os nós da árvore são as células de entrada ou saída do padrão de modificação e cada arco liga uma célula à decomposição que a substituiu (Figura 2.4). A Figura 2.5 ilustra uma representação alternativa do processo conhecida como árvore de segmentação. Sua construção pode ser feita da forma *top-down* ou *bottom-up*, dependendo se o padrão é de refinamento ou de simplificação, respectivamente. Percorrendo a árvore, pode-se escolher o grau de refinamento desejado para o modelo através de um corte transversal.

Cada nível da árvore de segmentação gera um modelo que abrange todo o domínio com a mesma resolução. Cortes que possuam nós em resoluções diferentes abrangerão o domínio adaptativamente, ou seja, em resoluções diferentes ao longo do domínio.

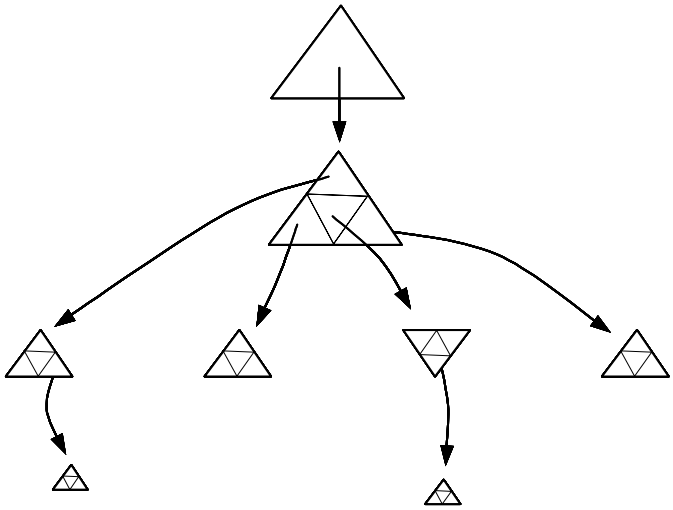


Figura 2.4 - Árvore que representa o processo de refinamento.

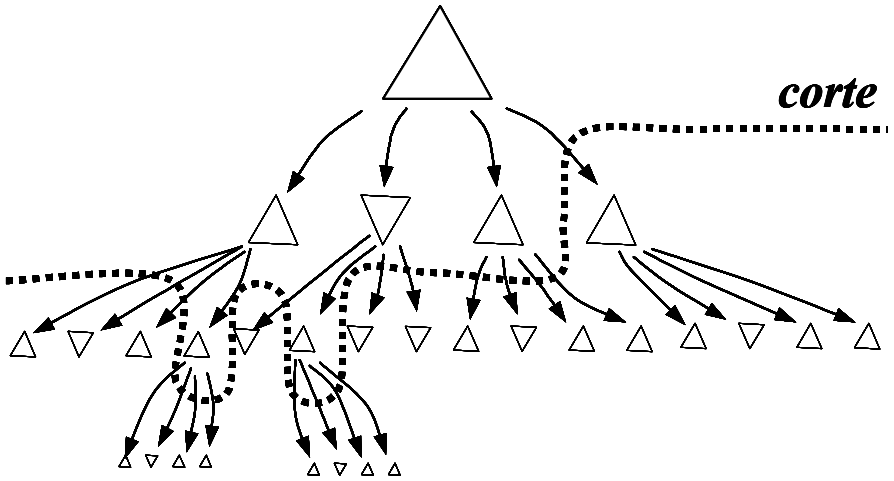


Figura 2.5 – Corte transversal na árvore.

O padrão de modificação é fundamental para a qualidade dos modelos gerados, principalmente quando a aplicação exige que os modelos sejam conformes. Neste caso, um padrão qualquer pode gerar modelos com descontinuidades entre as células, causadas pela formação de vértices T (Figura 2.6-a), principalmente entre células vizinhas de um mesmo corte que estão em níveis diferentes. Em superfícies imersas no $\Re^3$, as descontinuidades são evidenciadas pela presença de aberturas (*cracks*) (Figura 2.6-b).

Os modelos que possuem descontinuidades geradas por vértices T são não-conformes. A maioria das aplicações deve evitar a existência dessas descontinuidades. Para tanto, é necessário utilizar padrões de modificação adequados e criar regras que conduzam a escolha dos cortes a serem feitos durante o caminhamento na árvore. Neste sentido, as principais soluções propostas são árvores quaternárias com restrições [Her87], triangulações adaptativas [Flo95, Sca92], hierarquia de triângulos direitos [Heb95] e hierarquia de triângulos do tipo 4-8 [Vel00]. Elas variam quanto ao tipo de célula utilizada, ao padrão de modificação empregado, ao número de padrões disponíveis e às regras de caminhamento na hierarquia para gerar cortes.

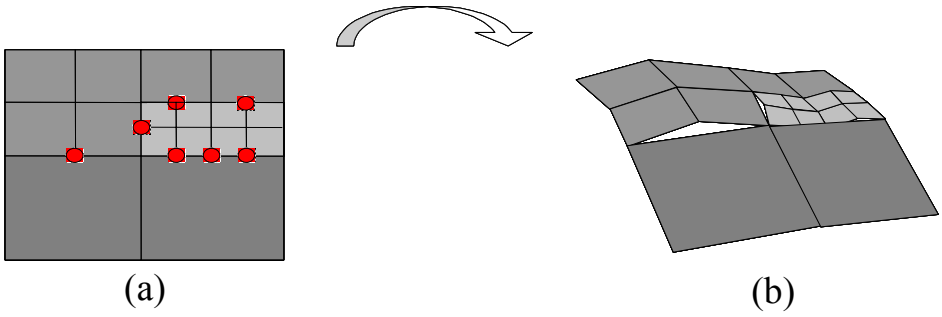


Figura 2.6 - Aberturas geradas por vértices T, em destaque.

Em geral, os MMRHs possuem boas propriedades, como indexação espacial [Pup97], agrupamento com coerência espacial (*clustering*) [Gar98] e simplicidade de implementação e entendimento.

Por outro lado, a geração do modelo base pode não ser trivial no caso de refinamento. Além disso, a presença de aberturas é um problema sério, cujas soluções normalmente possuem algum tipo de efeito colateral indesejado, como:

- a geração de modelos com células com uma péssima razão de aspecto, fato ressaltado por Floriani [Flo84] que se deve à tentativa de manter intactas as arestas das células ao longo de diferentes níveis;

- a redução do poder de expressão da hierarquia (ex. [Her87]), uma vez que os cortes válidos são restringidos; e

- o aumento do tempo de extração de um modelo, devido a várias passadas na hierarquia para tornar o modelo conforme (ex. [Her87]).

As soluções mais promissoras parecem estar relacionadas à hierarquia de triângulos do tipo 4-8 tratada por Velho [Vel00, Vel01, Vel01b]. Entretanto, ainda não é claro como esta solução pode ser aplicada a malhas não-conformes.

2.3 Modelos de Multi-resolução Reticulares

Ao contrário dos modelos hierárquicos, os modelos de multi-resolução reticulares (**MMRRs**) não são dados por padrões de modificação fixos aplicados a cada célula individualmente. Dado um modelo base $\sum_0$, qualquer sub-modelo $\kappa_i \subseteq \sum_0$ pode sofrer uma transformação $\boldsymbol{T}$ gerando um novo sub-modelo κ_j, mais ou menos refinado, (Figura 2.7) que abrange o mesmo domínio de κ_i ($\boldsymbol{dom}(\kappa_j) = \boldsymbol{dom}(\kappa_i)$):

$$\boldsymbol{T}(\kappa_i) = \kappa_j.$$

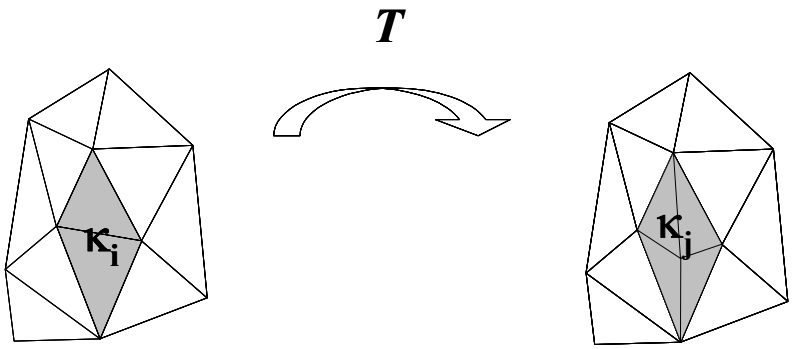


Figura 2.7 - Transformação de decomposição aplicada a um domínio.

A idéia é que uma série dessas transformações $\{\boldsymbol{T}_1(\kappa_1), \boldsymbol{T}_2(\kappa_2), ..., \boldsymbol{T}_n(\kappa_n)\}$, cuja união dos domínios pode potencialmente abranger todo o domínio $\boldsymbol{dom}(\sum_0)$, pode ocorrer em paralelo desde que os domínios sejam disjuntos entre si (ou seja, os domínios não interfiram uns nos outros). Após a aplicação de todas as transformações, o modelo troca globalmente de resolução, gerando um novo modelo $\sum_1$. O processo é repetido para o modelo $\sum_l$ e assim sucessivamente, até que o modelo com a resolução desejada seja alcançado.

Pode-se ilustrar o processo de modificação através da Figura 2.8. Para cada célula do modelo $\sum_i$, existem arcos direcionados ligando-a às células do complexo $\sum_{i+1}$ cujos domínios interceptam o domínio da célula em questão. Percebe-se que as células que permanecem inalteradas nas duas malhas formam arcos ditos **triviais**. Além disso, uma célula $\boldsymbol{c}_k \in \sum_{i+1}$ pode ter interseção com mais de uma célula de $\sum_i$, recebendo

mais de um arco. Este é o motivo que, ao contrário dos MMRHs, impede que este modelo seja representado por uma árvore.

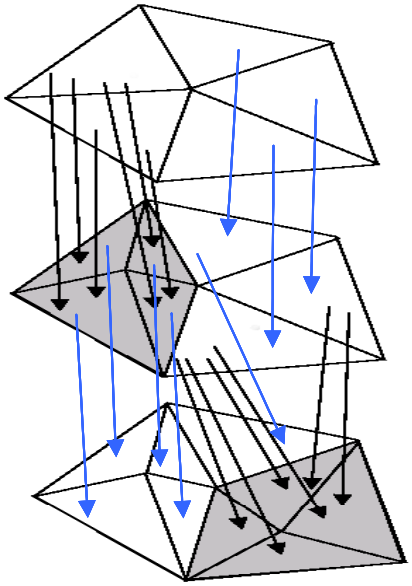


Figura 2.8 - Dependências entre domínios. As setas azuis representam dependências óbvias e são denominadas arcos triviais.

Uma condição importante é que as transformações ***T*** devem ser **operações locais**, ou seja, que alteram a conectividade e a geometria do modelo apenas localmente. Normalmente, tais modificações são baseadas em **operadores** que determinam como a modificação será realizada. Um operador aumenta ou diminui a complexidade do modelo gradualmente, podendo ser um operador de refinamento ou de simplificação, respectivamente. Os operadores mais comuns são: contração de arestas, contração de vértices, expansão de vértices, remoção de vértices, inserção de vértices, contração de células e expansão de células (Figura 2.9). Observando a Figura 2.9, percebe-se que a conectividade da borda da operação não é alterada. A exceção a esta regra são as bordas da operação que coincidem com as bordas do objeto, que podem ser alteradas.

É necessário enfatizar que a maioria dos padrões de modificação utilizados em MMRHs não são operadores e não são locais, de modo que eles normalmente alteram a conectividade e a topologia do modelo além do domínio no qual são aplicados.

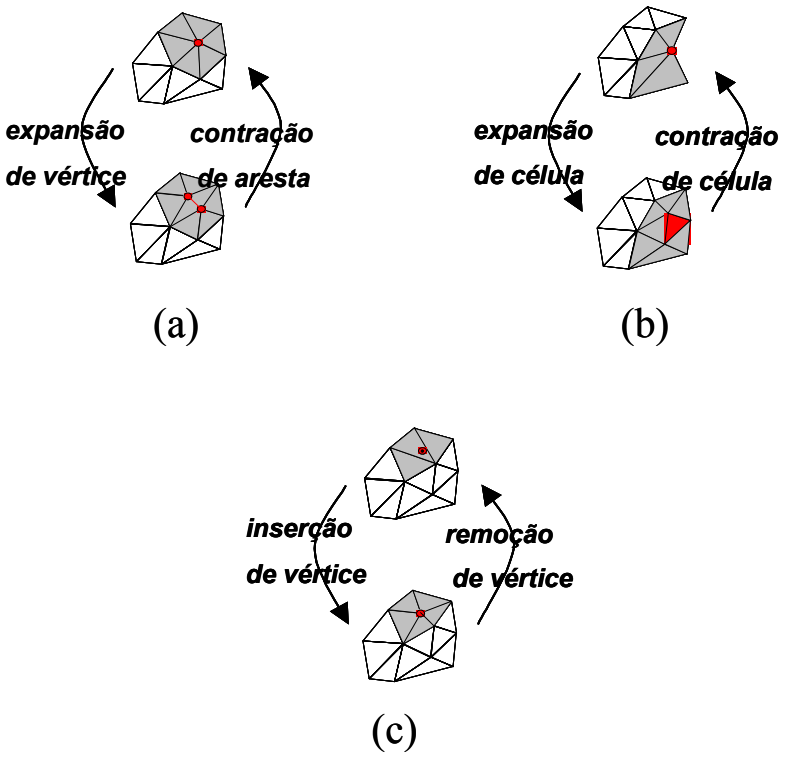


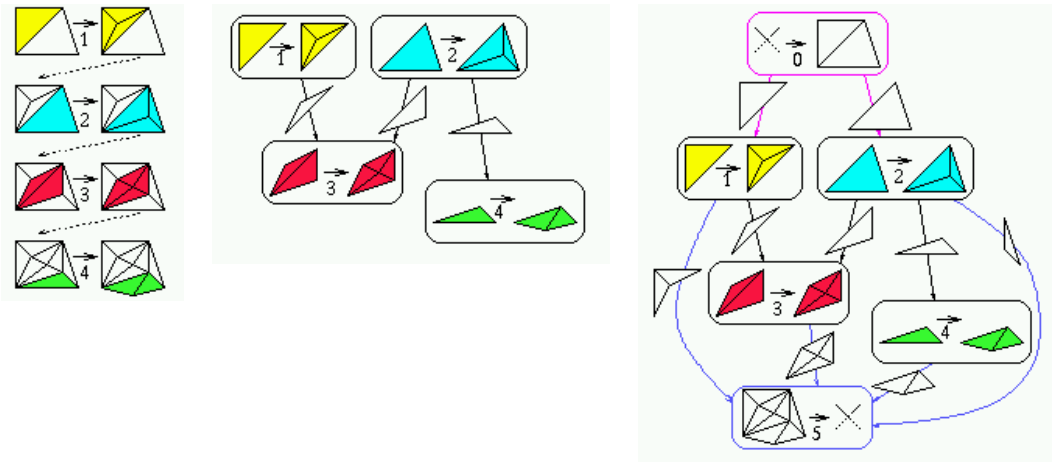
Figura 2.9 – Operações locais.

Os algoritmos que implementam operadores também são chamados de **incrementais** porque são capazes de criar um **histórico** cronológico de operações locais que alteram monotonicamente e gradualmente a complexidade do modelo. Como as operações podem ser tanto de refinamento quanto de simplificação, os históricos podem levar do modelo mais refinado ao mais simplificado ou o contrário.

Para se obter maior compactação e ter um tempo de extração proporcional ao tamanho do modelo extraído, é necessário utilizar históricos que contenham operações de refinamento. Um bom exemplo desta estratégia é o trabalho de Hoppe, que apresenta o conceito de malhas progressivas [Hop96].

Entretanto, apenas o uso de históricos não é suficiente para representar todo o modelo reticular. Para representar inteiramente um MMRR são necessários o histórico e as relações de interferência entre as operações. O trabalho seguinte de Hoppe, que trata da extração de malhas dependentes da câmera [Hop97], é um bom exemplo do uso de históricos juntamente com um estudo de análise de interferência.

Outro trabalho importante que, dentre outras contribuições, possibilita uma visão abrangente dos modelos reticulares foi o de Floriani, Puppo e Magillo [Flo97]. Nele, um MMRR é representado por um **grafo acíclico direcionado** (DAG). Os nós do grafo representam as operações do histórico e os arcos representam as relações de dependência diretas entre as operações efetuadas. Uma operação *a* depende diretamente de outra *b*, se *a* eliminar alguma célula criada por *b* (Figura 2.10), ou seja, $\boldsymbol{dom}(a) \cap \boldsymbol{dom}(b) \neq \varnothing$. Por outro lado, *a* depende indiretamente de *b* se existir um caminho no grafo que leve de *b* até *a*. A extração de modelos pode ser feita através de um caminhamento pelo grafo, desde que sejam respeitadas as relações de dependência. Com este novo caminhamento, o poder de expressão do MMR é maior do que o simples uso do histórico de operações, possibilitando a geração de malhas com diferentes resoluções ao longo do domínio. Uma vez que os operadores não alteram a conectividade na borda da operação, as malhas produzidas pelo caminhamento mantêm naturalmente a topologia das regiões *manifold*.



(a) ***(b)*** ***(c)***

Figura 2.10 – Modelo Reticular. Em (a), histórico (*H*) de operações de refinamento. Em (b), o DAG que representa o histórico e suas relações de dependência. Em (c), a inserção de dois *dummy nodes* que representam a malha mais refinada e a menos refinada.

3 Reservatórios Naturais de Petróleo e Requisitos de Visualização

O capítulo anterior proporcionou uma visão geral do problema de MR. Assim, convém, neste momento, apresentar uma definição mais detalhada das malhas de simulação em reservatórios que são o foco de estudo deste trabalho, objetivando instanciar melhor o problema em si. Para tanto, é necessário conhecer as características geométricas dessas malhas e os requisitos de sua visualização, uma vez que esta se diferencia da visualização foto-realista.

A Seção 3.1 descreve brevemente o que é um RNP e o que é a simulação numérica do fluxo de fluidos em reservatórios. Na Seção 3.2, são apresentados os dados provenientes da simulação, dentre eles as malhas de simulação e seus atributos. Finalmente, na Seção 3.3, os requisitos de visualização exigidos são detalhados.

3.1 Reservatório Natural de Petróleo

Segundo Gehardt [Ger98], a grande maioria das ocorrências de acumulações de petróleo (hidrocarbonetos como óleo e gás) se encontra em rochas sedimentares. Uma rocha sedimentar é formada a partir da acumulação, diagênese e solidificação de sedimentos, podendo ser de origem clástica, química ou orgânica. À medida que os sedimentos vão se empilhando em uma bacia sedimentar, o aumento de pressão ativa

os processos físicos, químicos e biológicos de formação de rochas sedimentares que compõem a diagênese, tais como: compactação, cimentação, recristalização, hidratação, lixiviação, ação de bactérias, etc. A uma temperatura adequada, os sedimentos orgânicos podem começar a gerar hidrocarbonetos, sendo então denominados rochas geradoras. Os hidrocarbonetos fluem através de meios permeáveis segundo o fenômeno da migração. Por terem densidades menores que a da água, são por ela deslocados ao encontrar uma via de escape. Se nenhuma trapa (barreira) for encontrada, acabam por escapar pela superfície da terra. Entretanto, se durante a migração existir uma combinação entre uma rocha capeadora impermeável e um meio poroso, os hidrocarbonetos podem ser trapeados e acumulados, formando um reservatório. Meios porosos típicos são arenitos e calcários, enquanto os selantes mais comuns são chamados de rochas sedimentares não-permeáveis, como os folhelhos. A Figura 3.1 ilustra dois tipos de reservatórios, um associado com uma estrutura capeadora do tipo convexo (anticlinal) e outro com um deslocamento linear (**falha geológica** ou **falha**).

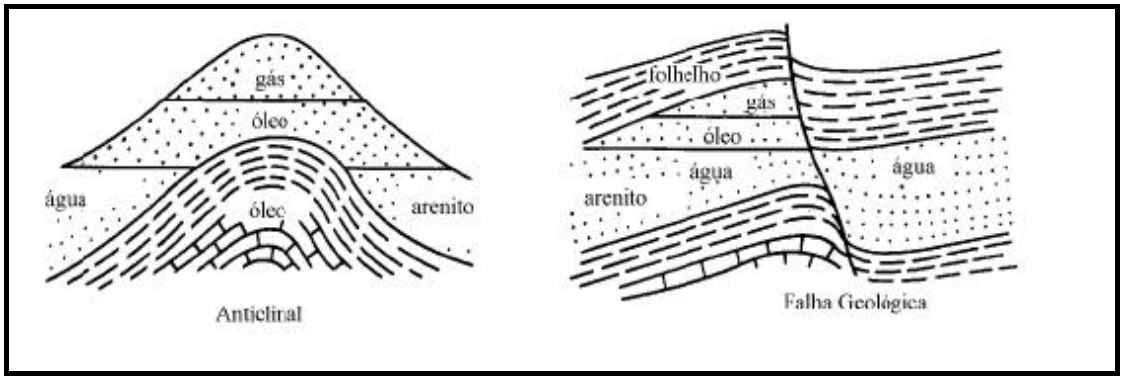


Figura 3.1 - Acumulações de hidrocarbonetos: (a) estrutura do tipo anticlinal e (b) estrutura do tipo falha (Robison et al., 1980).

O processo exploratório de um RNP baseia-se na capacidade estimada de extração do petróleo [Tho01]. Pode-se calcular esta capacidade através de simuladores numéricos baseados em estimativas prévias das propriedades das rochas e dos fluidos obtidas pela geologia e pela geofísica. Essas estimativas prévias são reajustadas à medida que se obtêm medidas de vazão e pressão dos poços.

O escoamento dos hidrocarbonetos dentro da camada permeável é modelado matematicamente por equações diferenciais de fluxos em meios porosos. Tais

equações são normalmente resolvidas numericamente representando o seu domínio através de uma malha discreta de células hexaédricas (malha de simulação) às quais são atribuídos os valores das propriedades físicas.

O modelo geométrico do reservatório é composto pelo conjunto de células. A quantidade e o tamanho dessas células variam de acordo com os requisitos da simulação numérica. Ou seja, há uma importante diferença em relação aos modelos utilizados em animações e na representação de ambientes virtuais. No caso destes, a resolução é determinada pela necessidade de se obter imagens realistas dos modelos, enquanto, no caso de malhas de simulações numéricas, a resolução é controlada pela necessidade de precisão dos valores das grandezas físicas estimadas.

3.2 Dados da Simulação

Os dados provenientes da simulação numérica correspondem à malha de simulação e às propriedades escalares (grandezas físicas) a ela associadas. Esta seção tem por objetivo detalhar esses dados a fim de possibilitar um melhor entendimento sobre o objeto que é o foco do MMR proposto neste trabalho.

3.2.1 Malhas de Simulação

Como foi visto brevemente no capítulo introdutório, as malhas de simulação são compostas por um conjunto de células hexaédricas indexadas por triplas **<*i, j, k*>** que associam às células uma suposta relação de vizinhança 6-conectada (**vizinhança IJK**): a célula **<*i, j, k*>** é vizinha das células **<*i*+1, *j, k*>** e **<*i*-1, *j, k*>**, em ***I*,** das células **<*i, j*+1, *k*>** e **<*i, j*-1, *k*>**, em ***J***, e das células **<*i, j, k*+1>** e **<*i, j, k*-1>,** em ***K***.

Cada célula é homeomorfa ao cubo e esta propriedade, acrescida da relação de vizinhança, faz com que, intuitivamente, o reservatório possa ser representado parametricamente a partir da partição regular de um cubo. O espaço paramétrico (dado pelos eixos ***i***, ***j*** e ***k***) auxilia na compreensão do dado, principalmente de sua

suposta vizinhança (Figura 3.2). A cada célula do espaço paramétrico é aplicado um homeomorfismo ***f*** que leva a célula para o espaço geométrico (dado pelos eixos ***x***, ***y*** e *z*).

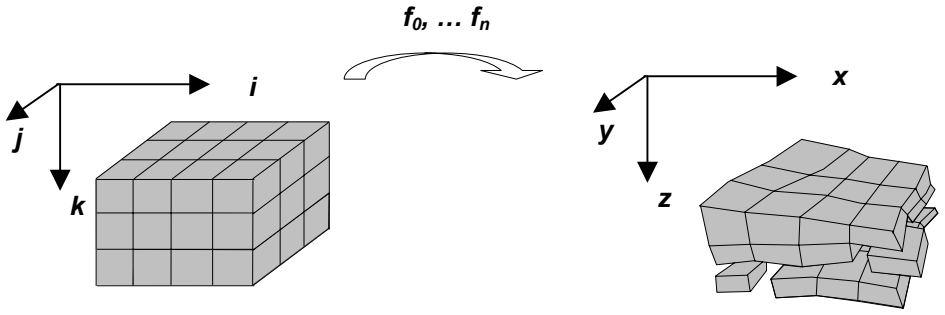


Figura 3.2 – Representação paramétrica das malhas de simulação do fluxo de fluidos em reservatórios.

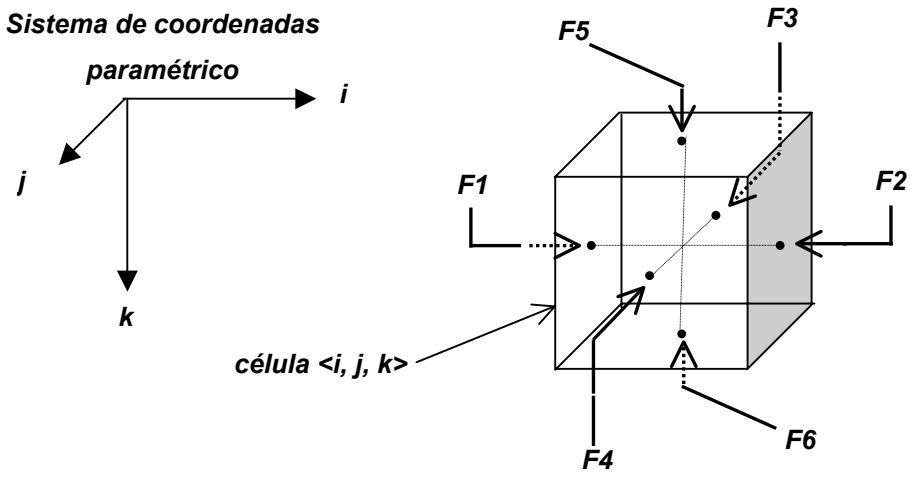


Figura 3.3 - Topologia de uma célula. *F1* está à esquerda de *F2* segundo o eixo i. Da mesma forma, *F3* está à esquerda de *F4* segundo o eixo j e *F5* está à esquerda de *F6* segundo o eixo k. A face *F5* é também é denominada *TOPO*, a face *F6* é a *BASE* e o conjunto de faces { *F1, F2, F3, F4* } é denominado *CERCA*.

No espaço paramétrico, o reservatório está alinhado com os eixos ***i***, ***j*** e ***k*** e cada célula é composta por faces que são identificadas através de rótulos, como na Figura 3.3. Assim, se uma célula ***A*** está à esquerda de uma célula ***B***, em ***I***, significa que $A \cap B$ corresponde à face F2 de ***A*** (que é igual a F1 de ***B***). As faces F2 e F1 de ***A*** e ***B***, respectivamente, são denominadas **faces potenciais de contato** entre ***A*** e ***B***. Analogamente, as faces potenciais de contato são definidas nas outras direções, ***J*** e ***K***.

Considerando a representação paramétrica, a geometria dos dados pode ser representada pelo conjunto das n células $\boldsymbol{C} = \{\boldsymbol{c_1}, \boldsymbol{c_2}, \boldsymbol{c_3}, ..., \boldsymbol{c_n}\}$ e pelo conjunto dos n homeomorfismos $\boldsymbol{\xi} = \{\boldsymbol{f_1}, \boldsymbol{f_2}, \boldsymbol{f_3}, ..., \boldsymbol{f_n}\}$ que levam as células do espaço paramétrico para o espaço geométrico. Assim, $\boldsymbol{G} = \{\boldsymbol{C}, \boldsymbol{\xi}\}$ é o conjunto que representa a geometria do reservatório.

A relação de vizinhança está implícita na indexação linearizada de cada célula, uma vez que é conhecida a quantidade máxima de células em cada direção (**MAX_I**, **MAX_J** e **MAX_K**) e a fórmula de linearização. Entretanto, explicitando, pode-se representar a vizinhança através de um grafo, $\boldsymbol{A} = (\boldsymbol{N}, \boldsymbol{Arc})$, denominado **grafo de adjacência**, onde $\boldsymbol{N}$ é o conjunto de nós do grafo que representa as células do reservatório e $\boldsymbol{Arc}$ é o conjunto de arcos não-direcionados que representa a vizinhança IJK (Figura 3.4).

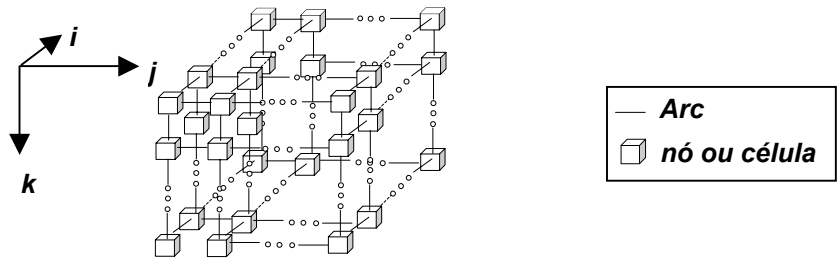


Figura 3.4 - Grafo de adjacência.

Para a simulação, é importante que as faces potenciais de contato entre duas células possuam realmente uma superfície de contato no espaço geométrico. A existência desses contatos reais, além daqueles verificados pelo caminhamento em $\boldsymbol{K}$, torna possível a passagem de fluido entre células. Tendo em mente a representação paramétrica do reservatório, o conjunto $\boldsymbol{G}$ e o grafo $\boldsymbol{A}$, são caracterizados dois tipos de malhas de simulação que descrevem com maior ou menor precisão a possibilidade de fluxo entre as células e, da mesma forma, se adaptam melhor ou pior à geometria do reservatório no mundo real e aos métodos numéricos empregados na simulação: as malhas do tipo *block-centered* e as *corner-point*. Cada tipo de malha, de certa forma, impõe restrições que correlacionam os homeomorfismos de $\boldsymbol{\xi}$, garantindo a existência de áreas de contato entre as células.

3.2.1.1 Malhas do Tipo Block-centered

As restrições impostas à geometria das malhas *block-centered* podem ser percebidas através da forma com que a malha é fornecida pelo simulador. Predefinindo-se que as células que a formam são alinhadas com os eixos do sistema de coordenadas, a malha é fornecida através de dois vetores, **DX** e **DY**, e de duas matrizes tridimensionais, **ZTOPO** e **ZBASE**[6].

Os vetores **DX** e **DY** são, respectivamente, funções dos indexadores ***i*** e ***j***. **DX** define o comprimento, em ***X***, das células indexadas por coordenadas **<*i***, ∗, ∗**>** e, analogamente, **DY** define o comprimento, em ***Y***, das células indexadas por **<**∗, ***j***, ∗**>**. Estes comprimentos (cada um para seu indexador), dispostos em seqüência, formam duas discretizações: uma do eixo ***x*** e outra do eixo ***y*** (Figura 3.5).

O produto cartesiano dessas duas discretizações forma uma discretização regular de um suporte retangular $U \subset \Re^2$, que é um subconjunto do plano formado pelos eixos ***x*** e ***y*** (Figura 3.6). Esta discretização induz, naturalmente, a uma subdivisão do plano em células planares retangulares que reconstrói o suporte ***U***, a qual convém chamar de **malha de referência**. (Figura 3.7).

As matrizes **ZTOPO** e **ZBASE** completam as informações existentes na malha de referência, atribuindo a cada célula <***i***, ***j***, ***k***> do reservatório as posições, em ***Z***, de sua face **TOPO** e de sua face **BASE**, respectivamente (Figura 3.8).

---

[6] *Na verdade, o simulador denomina estas duas matrizes de* **TOPO** *e* **BASE**. *As denominações* **ZTOPO** *e* **ZBASE** *são feitas para distingui-las das denominações* **TOPO** *e* **BASE** *atribuídas às faces F5 e F6.*

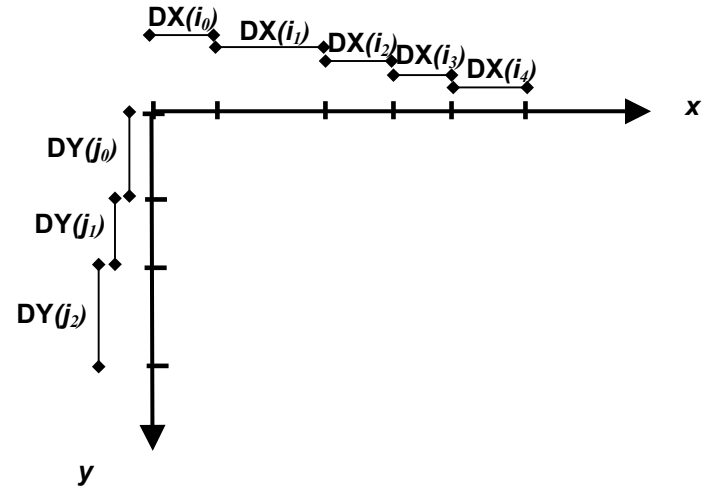


Figura 3.5 - Discretizações dos eixos *x* e *y*.

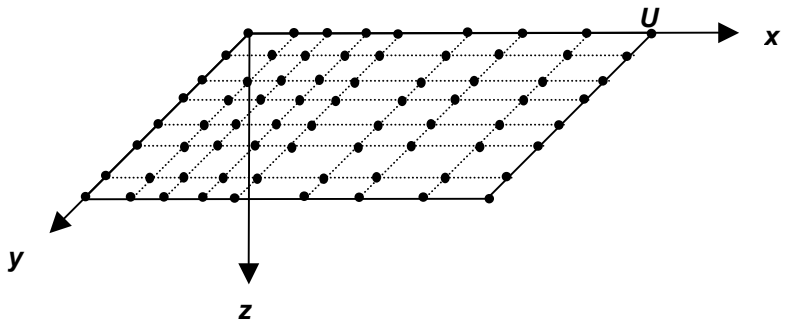


Figura 3.6 – Produto cartesiano das discretizações dos eixos *x* e *y*.

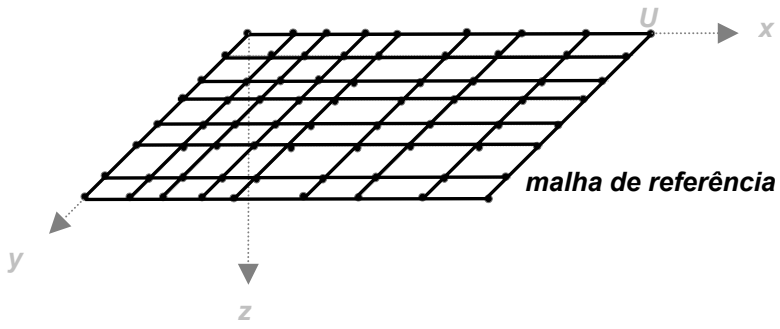


Figura 3.7 - Malha de referência que reconstrói o suporte *U*.

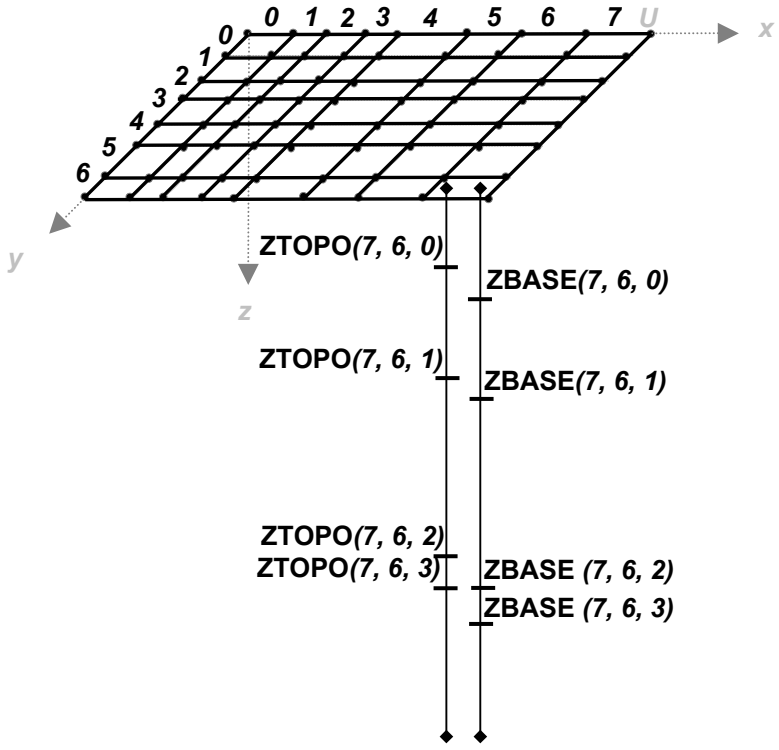


Figura 3.8 – Informações fornecidas por ZTOPO e ZBASE para as células <7, 6, 0>, <7, 6, 1>,<7, 6, 2> e <7, 6, 3>.

A partir de **DX** e **DY**, tem-se diretamente o comprimento das células em ***X*** e em ***Y*** e, a partir de **ZTOPO** e **ZBASE**, tem-se o comprimento das células em ***Z*** (subtraindo-se **ZTOPO**(***i***, ***j***, ***k***) de **ZBASE**(***i***, ***j***, ***k***)). As translações que posicionam as células no espaço são derivadas de **DX**, **DY** e **ZTOPO**.

A geometria de uma célula <***i***, ***j***, ***k***> da malha pode, portanto, ser assim definida (Figura 3.9):

- **DX**(***i***) é seu comprimento na direção ***X***;
- **DY**(***j***) é seu comprimento na direção ***Y***;
- **DK**(***i***, ***j***, ***k***) = **ZBASE**(***i***, ***j***, ***k***) - **ZTOPO**(***i***, ***j***, ***k***) é seu comprimento na direção ***Z***;
- **T**($\sum_{i'=0}^{i-1} \mathrm{DX}(i')$, $\sum_{j'=0}^{j-1} \mathrm{DY}(j')$, **ZTOPO**(***i***, ***j***, ***k***)) é a translação que a posiciona no espaço;

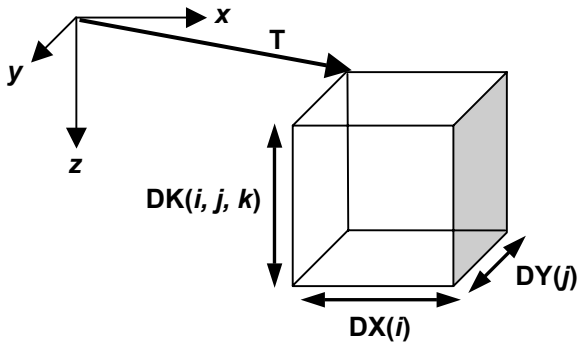


Figura 3.9 – Geometria de uma célula *block-centered*.

As restrições das malhas *block-centered*, garantidas por construção, impõem ao reservatório a uniformidade e a continuidade requeridas pela simulação para a transferência de fluidos. A restrição à malha de referência, por exemplo, garante que a vista superior (aquela perpendicular ao plano formado pelos eixos ***x*** e ***y***) de qualquer camada do reservatório seja idêntica à vista superior da malha de referência (Figura 3.10-a), implicando em que células indexadas por <***i***, ***j***, ∗> possuam suas faces F1 necessariamente contidas no mesmo plano (o mesmo vale para as faces F2, F3 e F4) (Figura 3.10-b). Conseqüentemente, células vizinhas em ***I***, ou em ***J***, possuem também

suas faces potenciais de contato contidas no mesmo plano no espaço geométrico (Figura 3.11).

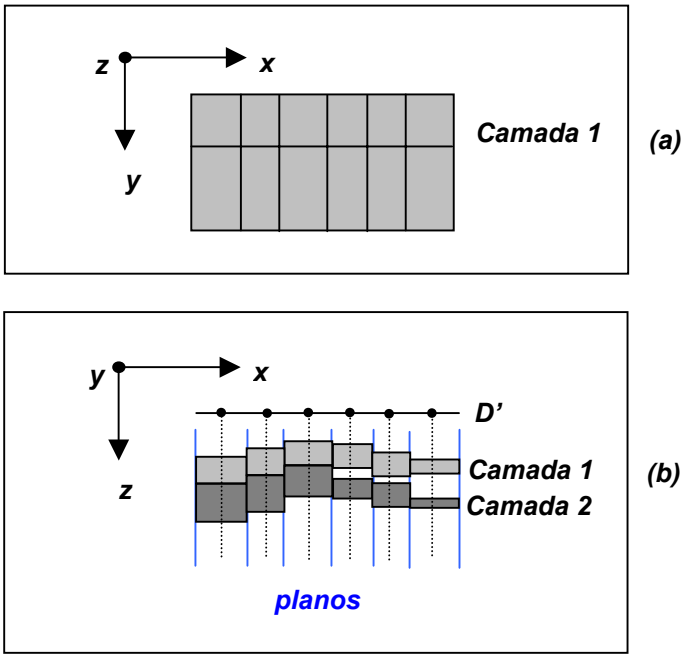


Figura 3.10 – Em (a), vista superior de uma camada. Em (b), vista lateral das células mostra que células indexadas por <*i*, *j*, *> possuem faces no mesmo plano.

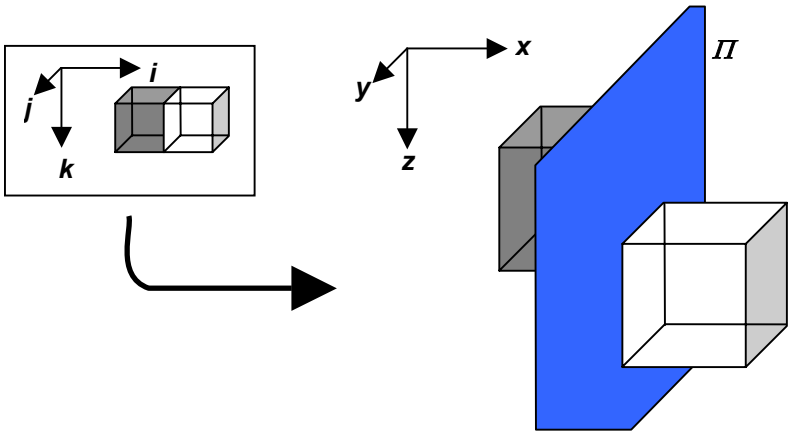


Figura 3.11 – Restrições das faces potenciais de contato.

A Figura 3.12 é um típico exemplo de malha do tipo *block-centered*.

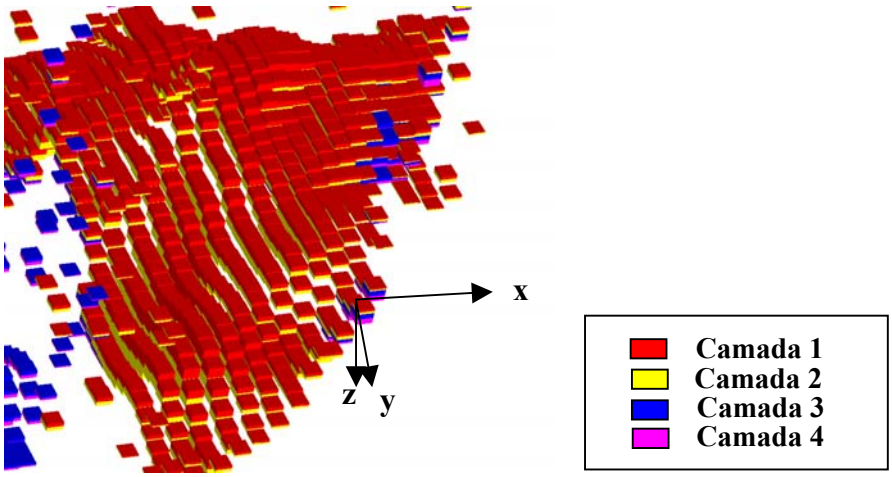


Figura 3.12 - Exemplo de uma malha do tipo block-centered (obs.: o eixo está deslocado).

3.2.1.2 Malhas do Tipo Corner-point

No espaço geométrico, uma malha do tipo *block-centered* possui células alinhadas com os eixos do sistema de coordenadas. Apesar da representação fácil e compacta, a malha do tipo *block-centered* apresenta muitas descontinuidades, denominadas **degraus** (Figura 3.13), e não se adaptam bem a geometrias complicadas, prejudicando, muitas vezes, a precisão das simulações.

Uma malha do tipo *corner-point* é uma generalização da malha do tipo *block-centered*. Ela não possui a limitação de ter células alinhadas com os eixos, possibilitando uma melhor representação da geometria típica de um terreno no mundo real. A conseqüência disto é a possibilidade de uma simulação mais precisa e realista do fluxo no RNP, desde que as células não apresentem, em sues vértices, ângulos muito distintos dos ângulos dos vértices de um cubo (ou seja, deformações podem levar a aproximações numéricas não muito boas).

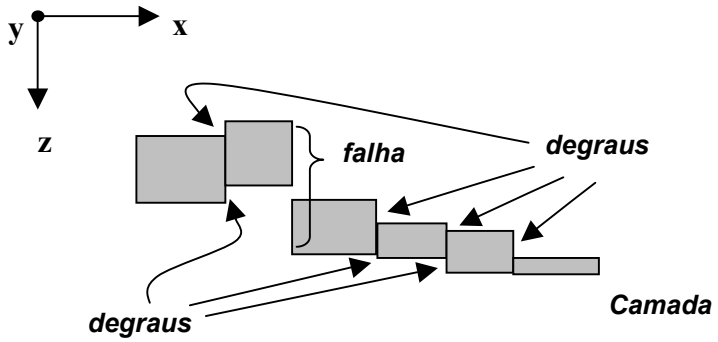


Figura 3.13 - Degraus e falhas em uma camada.

Cada célula da malha do tipo *corner-point* precisa ser explicitamente definida pelos seus oito vértices e pela indireção de suas faces, as quais são identificadas pelos rótulos F1, F2, F3, F4, F5 e F6. A geometria de uma face é definida pela interpolação bilinear de seus vértices e, portanto, não precisa ser planar. Entretanto, quanto mais próximas as faces forem de retângulos, melhores serão os resultados numéricos [IMEX00].

O fluxo pode existir em duas situações: (a) quando as faces potenciais de contato entre duas células compartilham os mesmos vértices no espaço geométrico (Figura 3.14-a) e (b) quando as seguintes condições são satisfeitas:

- as faces potenciais de contato são planares no espaço geométrico;

- as arestas ***direita*** (ver Figura 3.14-b) de cada face são colineares, assim como suas arestas ***esquerda***, e há alguma interseção entre elas.

Uma malha *corner-point* é fornecida pelo simulador diretamente através das coordenadas dos oito vértices de cada célula. Portanto, ao contrário das malhas do tipo *block-centered*, a forma com que as malhas *corner-point* são fornecidas não indica a existência de obediência às restrições requeridas pela simulação para a existência de fluxos.

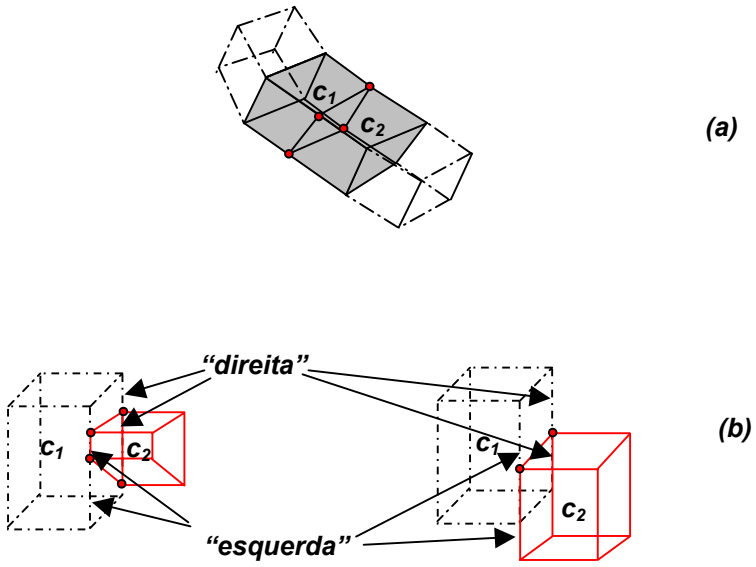


Figura 3.14 – Existência de fluxo entre células. Em (a), as faces potenciais de contato entre c_1 e c_2 compartilham os mesmos vértices. Em (b), as arestas *direita* de cada célula são colineares, assim como as arestas *esquerda*. Além disso, as faces potenciais de contato são coplanares.

Entretanto, tipicamente, essas malhas são geradas de forma que as condições de existência de fluxos possam ser satisfeitas. A geração é feita de forma semelhante ao modo com que a malha *block-centered* é fornecida pelo simulador. Para gerar a malha, é utilizada uma segunda malha de referência que forma uma decomposição em células quadrilaterais de uma região $U' \subset \Re^2$ contida no plano formado pelos eixos x e y (Figura 3.15). Às células desta malha é atribuída uma identificação baseada na indexação em I e em J (Figura 3.16).

A partir de cada vértice da malha de referência são definidas retas que conterão os vértices finais das células da malha de simulação. Tais retas não precisam ser exatamente perpendiculares ao plano que contém a malha de referência (Figura 3.17). Então, uma célula $<i, j, k>$ do reservatório é definida a partir da célula $<i, j>$ da malha de referência da seguinte forma: para cada reta que parte de um vértice da célula

<***i***, ***j***>, são atribuídos valores de topo e de base, definindo os vértices finais da célula <***i***, ***j***, ***k***> (Figura 3.18).

Comparando visualmente os dois tipos de malhas de simulação (Figura 3.19), percebe-se a diferença entre elas. Devido à flexibilidade na definição da geometria das células, malhas do tipo *corner-point*, normalmente, formam decomposições hexaédricas de sub-regiões do reservatório (potencialmente, uma sub-região pode ser o próprio terreno como um todo), ajustando-se muito bem à geometria da sub-região no mundo real sem possuir muitos degraus.

Um exemplo típico de uma malha do tipo *corner-point* é ilustrado pela Figura 3.20.

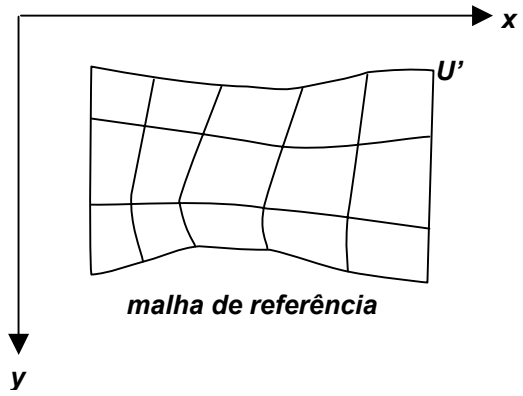


Figura 3.15 - Malha de referência que reconstrói o suporte *U'*.

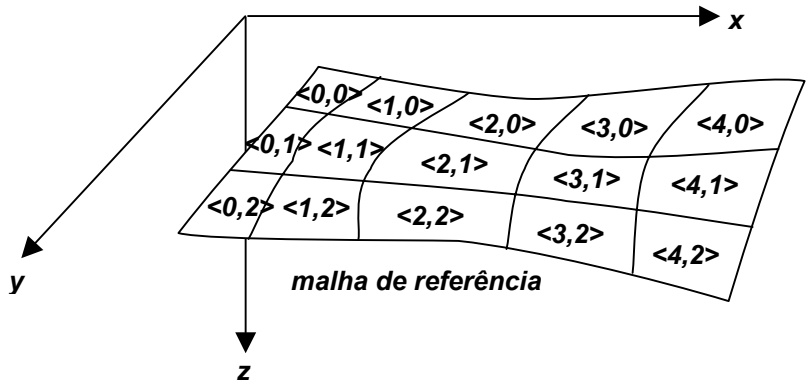


Figura 3.16 - Indexação das células da malha de referência.

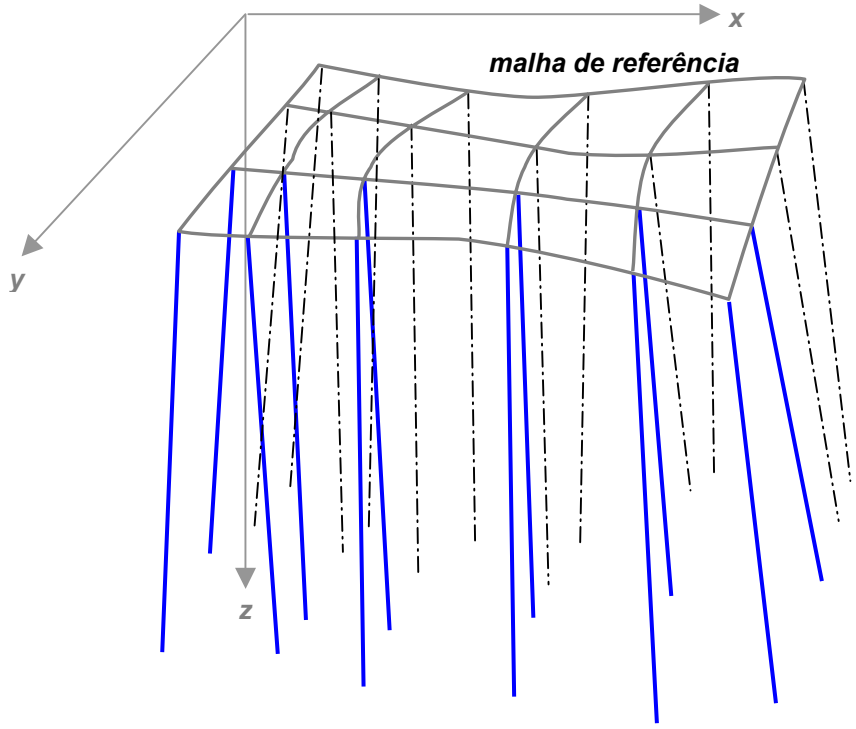


Figura 3.17 - Retas que passam pelos vértices da malha de referência.

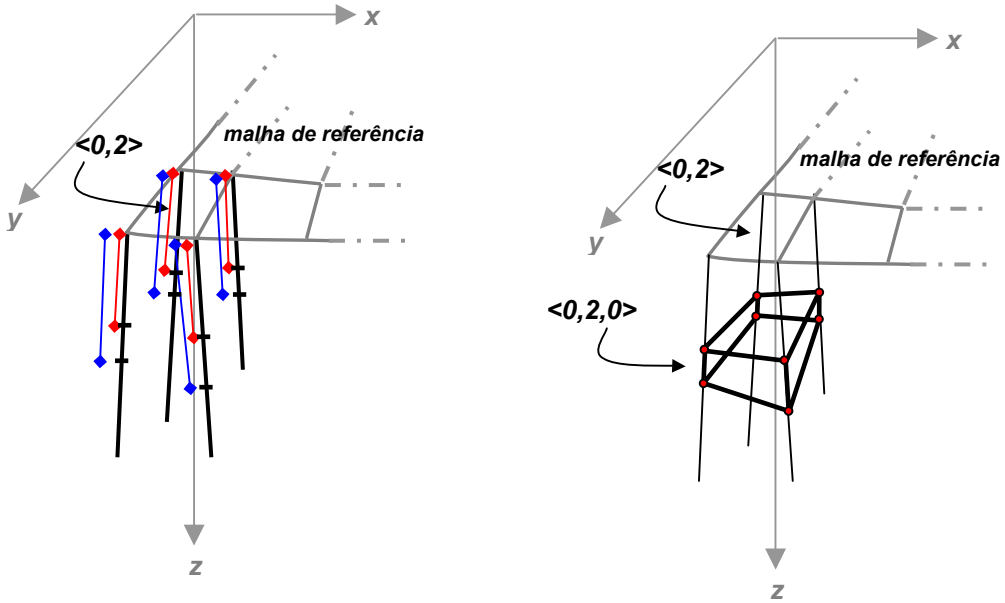


Figura 3.18 - Geração da célula <0, 2, 0>. À esquerda, em vermelho, estão representados os valores de topo de cada reta e, em azul, os valores de base. A figura à direita representa a célula final.

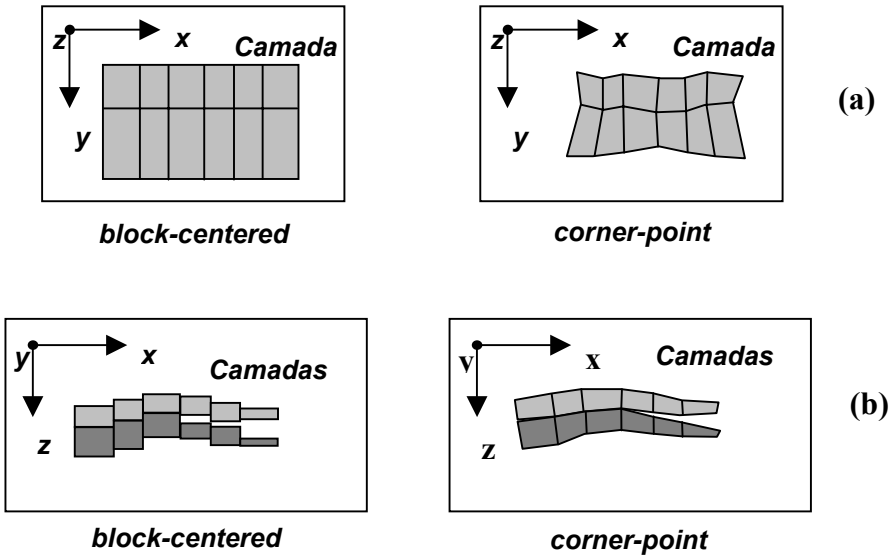


Figura 3.19 - Comparação entre os dois tipos de malhas. Em (a), vista superior de uma camada. Em (b), vista lateral de duas camadas.

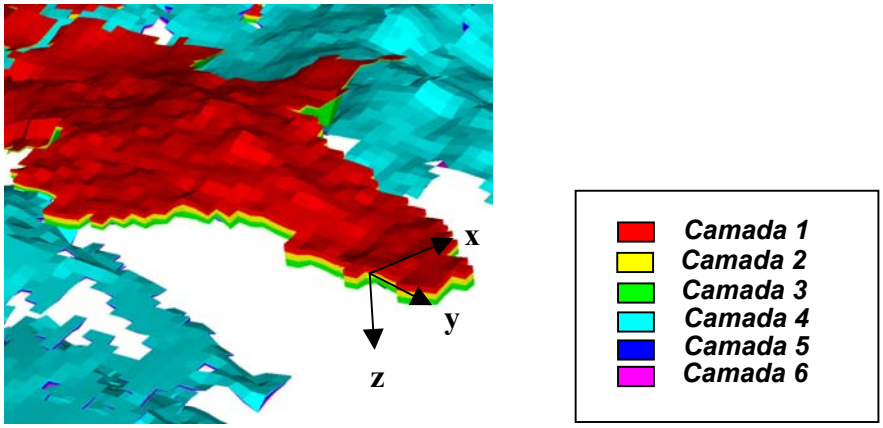


Figura 3.20 - Exemplo de uma malha do tipo *corner-point*.

3.2.2 Propriedades Escalares

As propriedades físicas que são associadas a cada célula do reservatório também fazem parte dos dados provenientes da simulação, representando grandezas específicas. Se as áreas de contato entre células tornam possível a existência de fluxos, o estado das propriedades do RNP, tais como densidade, porosidade e permeabilidade, define esses fluxos dentro do reservatório. Tais propriedades são medidas, ou estimadas, ao longo do reservatório em uma unidade de medida própria para cada uma. Dessa forma, valores escalares são associados a cada célula da geometria para representar as propriedades no interior da mesma.

Para definir formalmente as propriedades, consideremos o conjunto de células $\boldsymbol{C} = \{\boldsymbol{c_1}, \boldsymbol{c_2}, \boldsymbol{c_3}, ..., \boldsymbol{c_n}\}$. Uma propriedade do reservatório é definida através da função injetiva $\boldsymbol{i}:\boldsymbol{C}\rightarrow\Re$, que associa a cada célula $\boldsymbol{c_i}$ um valor escalar. Propriedades representadas dessa forma não variam de estado ao longo do tempo e são denominadas **mapas de inicialização**. Os mapas de inicialização são representados pelo conjunto $\boldsymbol{I} = \{\boldsymbol{i_1}, \boldsymbol{i_2}, \boldsymbol{i_3}, ..., \boldsymbol{i_p}\}$, $p \in$ N, onde cada $\boldsymbol{i_i}$ representa uma função e, logo, uma propriedade.

Por sua vez, as propriedades que variam em função do tempo, denominadas **mapas de recorrência**, são representadas através do conjunto $\boldsymbol{R}=\{\boldsymbol{r_1}, \boldsymbol{r_2}, \boldsymbol{r_3}, ..., \boldsymbol{r_p}\}$, onde cada $\boldsymbol{r_i}$ é uma função $\boldsymbol{r}: \boldsymbol{C}$ x N$\rightarrow\Re$, sendo N o conjunto dos números naturais que representa uma quantização do tempo.

Finalmente, chega-se à representação final dos dados provenientes da simulação através do conjunto $\{\boldsymbol{G}, \boldsymbol{S}\}$, onde $\boldsymbol{S} = \{\boldsymbol{I}, \boldsymbol{R}\}$, que encerra em si todos os dados que representam o comportamento estimado do reservatório ao longo dos anos.

Vale ressaltar a existência de um mapa inicial especial, denominado ***INAT***, que não está associado a grandezas físicas. Ele contém somente valores booleanos que indicam se uma célula é **ativa** ou **inativa**. Uma célula inativa, por definição, não possibilita o fluxo de fluidos[7]. Este mapa torna-se importante para o MMR a ser

[7] *Na verdade, uma célula inativa não faz parte do reservatório propriamente dito.*

proposto no Capítulo 4 porque tanto a geometria quanto os valores de grandezas físicas associados a uma célula inativa devem ser desconsiderados na visualização.

3.3 Requisitos de Visualização

O objetivo da visualização do conjunto $\{G, S\}$ é oferecer um meio eficiente de aquisição de informações sobre os dados oriundos da simulação[8]. Analisando as imagens desses dados, o engenheiro de petróleo pode inferir rapidamente a respeito do fenômeno ou, até mesmo, sobre a precisão da simulação realizada.

As técnicas de visualização utilizadas para gerar as imagens com a quantidade de informações exigida formam as **características de visualização** do problema em questão. Ao longo dos anos, os engenheiros de petróleo identificaram o conjunto de técnicas considerado adequado para formar as ditas características. Repensar estas técnicas não é a intenção deste trabalho. Assim, o modelo de multi-resolução proposto deve levar em consideração as características de visualização existentes, visando satisfazê-las.

Dentre as características existentes, as mais significativas são:

1. visualização da fronteira das células com, ou sem, seu *wireframe*, ou seja, não é necessário visualizar o interior de uma célula (Figura 3.22);

2. visualização por rótulo de faces, ou seja, pode-se escolher, alternativamente, a visualização somente da **BASE** ou do **TOPO** das células;

3. deformação arbitrária da geometria dos dados na direção **Z** feita em tempo de exibição;

4. afastamento arbitrário entre camadas de sedimentação (grupo de células com mesmo valor para a coordenada *k* – ver Figura 1.3);

---

[8] *Estes objetivos caracterizam a sub-área da Computação Gráfica denominada Visualização Científica (VC) [Cignoni98].*

5. latência pequena para iniciar a visualização após a leitura dos dados de simulação;

6. interatividade em tempo real durante a visualização;

7. visualização das propriedades através da associação de uma paleta de cores;

8. possibilidade de troca de mapas sem perda significativa de interação;

9. possibilidade de animação temporal dos mapas recorrentes sem perda significativa de interação.

A característica 1 é bastante importante não só porque elimina a necessidade de visualização do interior das células como também porque impõe definitivamente a semântica da célula hexaédrica para a visualização, uma vez que o *wireframe* das células pode ser requisitado (até agora a semântica somente era forte para a simulação).

A característica 2 enfatiza a mesma imposição anterior e também, juntamente com a característica 4, garante a importância das células internas da malha, pois aumenta a chance destas contribuírem para a imagem gerada.

A característica 5 é de fundamental importância, influenciando diretamente o MMR que será proposto no Capítulo 4. Ela determina a qualidade da heurística utilizada pelo algoritmo de construção do espaço discreto de modelos, aumentando a preocupação com o tempo gasto em pré-processamento (vale salientar que não é interessante para o usuário possuir um arquivo com a estrutura de MR pré-calculada, pois isso levaria à necessidade de lidar com mais um volume de dados).

Por sua vez, a característica 7 atribui uma semântica às propriedades dentro do contexto da visualização, uma vez que os valores escalares atribuídos a cada célula da malha serão utilizados como atributos de cor. Isto é feito através da associação de uma paleta de cores aos valores de propriedade (Figura 3.21).

Finalmente, a característica 6, além de ser a motivação principal deste trabalho, é também seu objetivo principal, tendo prioridade absoluta.

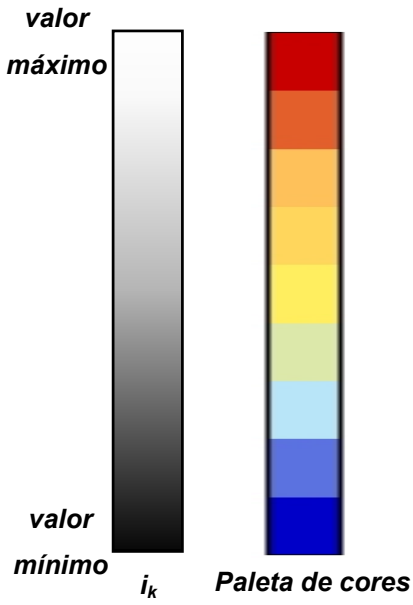


Figura 3.21 – Paleta de cores. A paleta exemplificada é discreta, mas pode ser utilizada também uma paleta contínua.

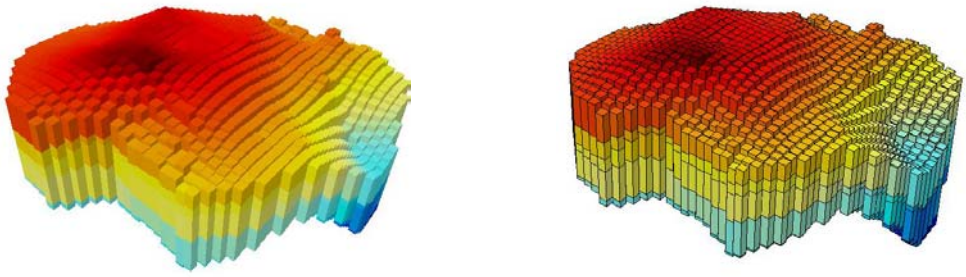


Figura 3.22 – Característica de visualização 1. À esquerda, a visualização da fronteira das células. À direita, a visualização da fronteira juntamente com o *wireframe* das células.

4 Algoritmo de Multi-resolução Proposto para RNPs

Este capítulo tem por objetivo propor um modelo de multi-resolução para RNPs, levando em consideração as características geométricas das malhas de simulação, a vizinhança IJK entre células, as propriedades escalares associadas a cada célula do reservatório e as características de visualização discutidas anteriormente.

O MMR é proposto na Seção 4.1, a Seção 4.2 detalha a operação local utilizada pelo modelo e a Seção 4.3 faz algumas considerações a respeito da estrutura de MR utilizada.

4.1 Hierarquia de Células Hexaédricas

Visando atender adequadamente às características de visualização descritas na Seção 3.4, o modelo de MR adotado é hierárquico, possuindo as características de indexação espacial e agrupamento com coerência espacial encontradas na maioria dos MMRHs, em particular, em [Gar98]. A célula hexaédrica foi utilizada como base do modelo proposto devido à sua importante semântica.

Para permitir a extração de malhas adaptativas sem causar a diminuição do poder de expressão do MMRH, são permitidas aberturas e pequenas interseções entre as células nas aproximações geradas. Isso pode alterar a topologia da malha de simulação

original, mas não é um fato crítico devido à existência típica de degraus e falhas nas malhas.

Apesar do algoritmo proposto se basear na vizinhança IJK, ele não segue uma hierarquia pré-definida, a exemplo da *octree*. As células são colapsadas em função do custo (erro) do colapso.

Inspirado nas técnicas de baixo para cima (*bottom-up*) incrementais de simplificação, normalmente relacionadas a MMRRs, o algoritmo de construção da estrutura hierárquica de multi-resolução aqui proposto consiste na escolha de uma série de operações incrementais que formam um histórico de operações, o qual, a exemplo das malhas progressivas [Hop96], leva do modelo mais refinado do reservatório até o seu modelo menos refinado. A partir deste histórico é possível identificar as dependências cronológicas entre operações e codificá-las através de uma árvore.

4.1.1 Colapso de Células

A operação utilizada pelo algoritmo proposto neste trabalho é uma operação de simplificação. Ela recebe duas células, $\boldsymbol{a}$ e $\boldsymbol{b}$, que abrangem a região do espaço ($\boldsymbol{a} \cup \boldsymbol{b}$), e retorna uma nova célula, $\boldsymbol{c}$, e o erro, $\boldsymbol{\varepsilon}$, associado à substituição de $\boldsymbol{a}$ e $\boldsymbol{b}$ por $\boldsymbol{c}$ (Figura 4.1). A esta aproximação no espaço geométrico está relacionada a propriedade no espaço paramétrico:

$$\boldsymbol{dom}(\boldsymbol{c}) = \boldsymbol{dom}(\boldsymbol{a} \cup \boldsymbol{b}) \text{ ,} \qquad \textbf{P.4.1}$$

A operação é denominada **colapso de células**, ou simplesmente **colapso**, e as células geradas por ele são denominadas **macro-células**.

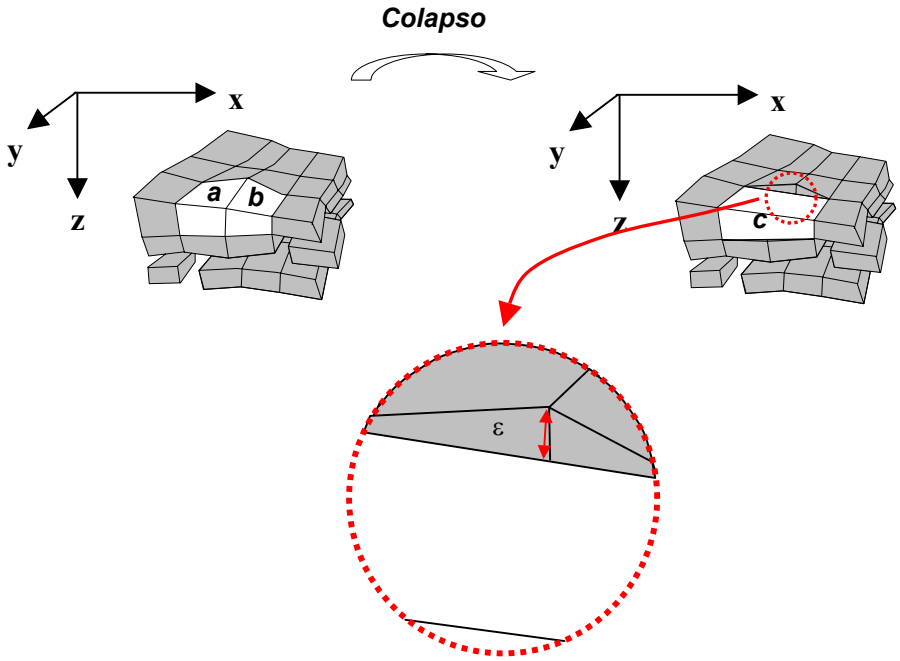


Figura 4.1 - Operação incremental de colapso. O colapso calcula a geometria da macro-célula, à direita, a partir das duas células, à esquerda. O erro ε, que nesta ilustração é o próprio erro geométrico, é associado ao colapso.

A Figura 4.2 mostra um exemplo bidimensional de abertura oriundo de uma operação de colapso.

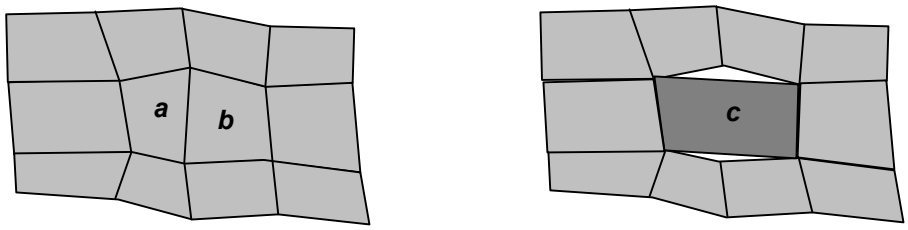


Figura 4.2 - Versão bidimensional da operação de colapso. A operação altera a borda de *(a ∪ b)* e, portanto, como neste exemplo, pode modificar a topologia fora do domínio ao qual é aplicada.

4.1.2 Algoritmo Guloso para Construção do Espaço de Modelos

O algoritmo proposto para a construção do espaço de modelos gerencia a construção de um histórico cronológico de operações de colapso, que são efetuadas uma a uma, levando a diferentes aproximações do reservatório que variam localmente uma da outra (Figura 4.3). Para criá-lo é utilizado um algoritmo guloso que iterativamente seleciona o colapso de menor erro.

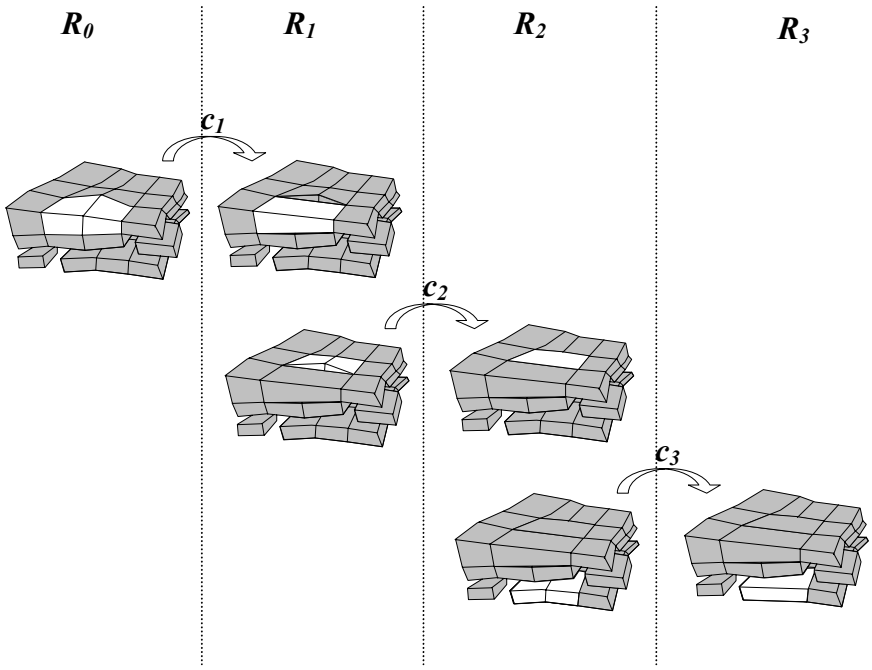


Figura 4.3 – Histórico. O histórico $\{c_1, c_2, c_3\}$ leva, a partir R_0, a mais três representações distintas do reservatório: R_1, R_2, R_3.

O algoritmo, em alto nível, pode ser especificado através da máquina de estado ilustrada na Figura 4.4. O estado inicial, E_0, é formado pelo reservatório original, R_0, e por um histórico vazio, ∅. Após a escolha do colapso de menor custo, a inserção deste no histórico e a efetuação do colapso, tem-se o próximo estado, E_1, que é formado por uma aproximação R_1 do reservatório original e por um histórico que, agora, contém um colapso a mais, $\{c_1\}$ (Figura 4.6). O algoritmo continua até que não existam mais colapsos a serem efetuados.

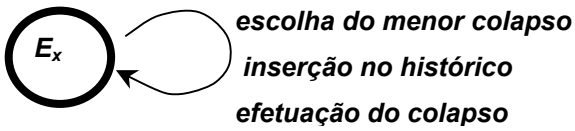


Figura 4.4 - Máquina de estado que representa o algoritmo guloso.

Dado um reservatório com n células, o número total de colapsos possíveis, sem considerar a vizinhança IJK, é n^2. Entretanto, considerando a vizinhança, que é explicitada pelo grafo de adjacência A, este número é próximo de $3n$, nunca excedendo-o. Uma vez que células vizinhas estão próximas no espaço geométrico, os colapsos entre elas possuem os menores erros, sendo provavelmente os mais relevantes para o algoritmo.

Assim, A pode ser interpretado de uma nova forma: seus nós continuam representando células, mas, agora, seus arcos representam colapsos e a cada arco está associado um custo que corresponde ao erro atribuído ao colapso.

Para extrair eficientemente de A o arco de menor custo, pode-se utilizar uma fila de prioridades, particularmente um *heap*, onde cada elemento é um arco. O *heap* pode ser criado em tempo linear no número de arcos a [Cor90]. Operações de extração do arco de menor custo e atualização de um elemento podem ser feitas em $O(\log a)$.

Devido à localidade da operação de colapso, a cada iteração do algoritmo, são necessárias apenas alterações locais nas estruturas de A e do *heap* para mantê-las consistentes com o próximo estado do algoritmo. A efetuação de um colapso faz com que o grafo de adjacência seja alterado e que um novo nó, que representa a macro-célula, seja criado, herdando a vizinhança dos nós colapsados (Figura 4.5). Assim, é necessário atualizar a vizinhança do novo nó recalculando custos. Isto pode ser feito em tempo proporcional ao grau g do novo nó. Além disso, é necessário tornar o *heap* consistente e atualizar os elementos cujos custos foram modificados. Isto pode ser feito em $O(g \log a)$.

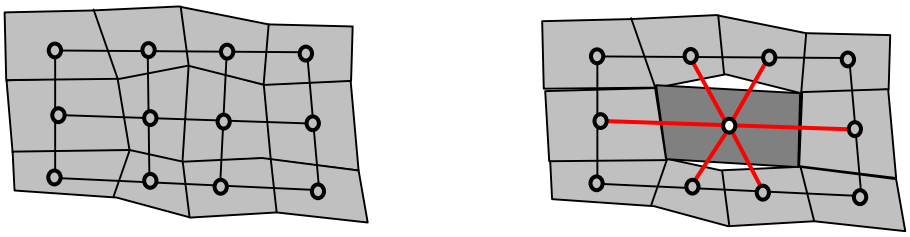


Figura 4.5 – Vizinhança. O novo nó herda a vizinhança dos nós colapsados. O grau do novo nó é igual à soma dos graus dos nós colapsados menos os colapsados que se tornaram redundantes menos 1.

Neste momento, vale fazer uma observação relativa à propriedade INAT, que indica se uma célula é ativa ou inativa. Como foi visto no Capítulo 3, a geometria e os valores de propriedade associados a uma célula inativa devem ser desprezados, fato este que levaria à existência de colapsos inócuos. Isso é evitado pelo algoritmo através de uma modificação no grafo ***A***: se uma célula é vizinha de uma célula inativa, esta vizinhança é alterada de forma que a nova vizinha seja a próxima célula ativa naquela direção. Dessa forma, alguns colapsos são eliminados e o número de arcos ***a*** do grafo passa a ser próximo a 3***n'***, onde ***n'*** é o número de células ativas.

O pseudo-código do algoritmo é o seguinte:

1° passo: *constrói o grafo de adjacência eliminando células inativas //O(**n**)*
2° passo: *constrói o heap //O(**a**)*
3° passo: *constrói o histórico vazio //O(1)*
4° passo: *enquanto houver colapsos*
{
    *extrai o menor colapso //O(log**a**)*
    *atualiza o grafo de vizinhança //O(**g**)*
    *atualiza o heap //O(**g**log**a**)*
    insere no histórico //O(1)
}

Como ***n*** difere de ***n'*** pela soma de uma constante e, uma vez que ***a*** é proporcional a ***n'***, a complexidade[9] final do algoritmo é O(***n'*** log ***n'***), desde que a variável ***g*** seja limitada superiormente por uma constante. Para isso, o algoritmo controla o grau dos

nós do grafo de adjacência durante a iteração. Assim, todo colapso que geraria um novo nó com grau acima de um valor limite ***G*** é proibido de ser efetuado, recebendo um custo elevado.

Cada colapso reduz em uma unidade o número de células. Logo, são efetuados ***n'***-1 colapsos sendo criadas exatamente ***n'***-1 macro-células. Isto implica em um histórico de complexidade O(***n'***).

---

[9] *A prova completa da complexidade deste tipo de algoritmo guloso pode ser encontrada em [Garland99].*

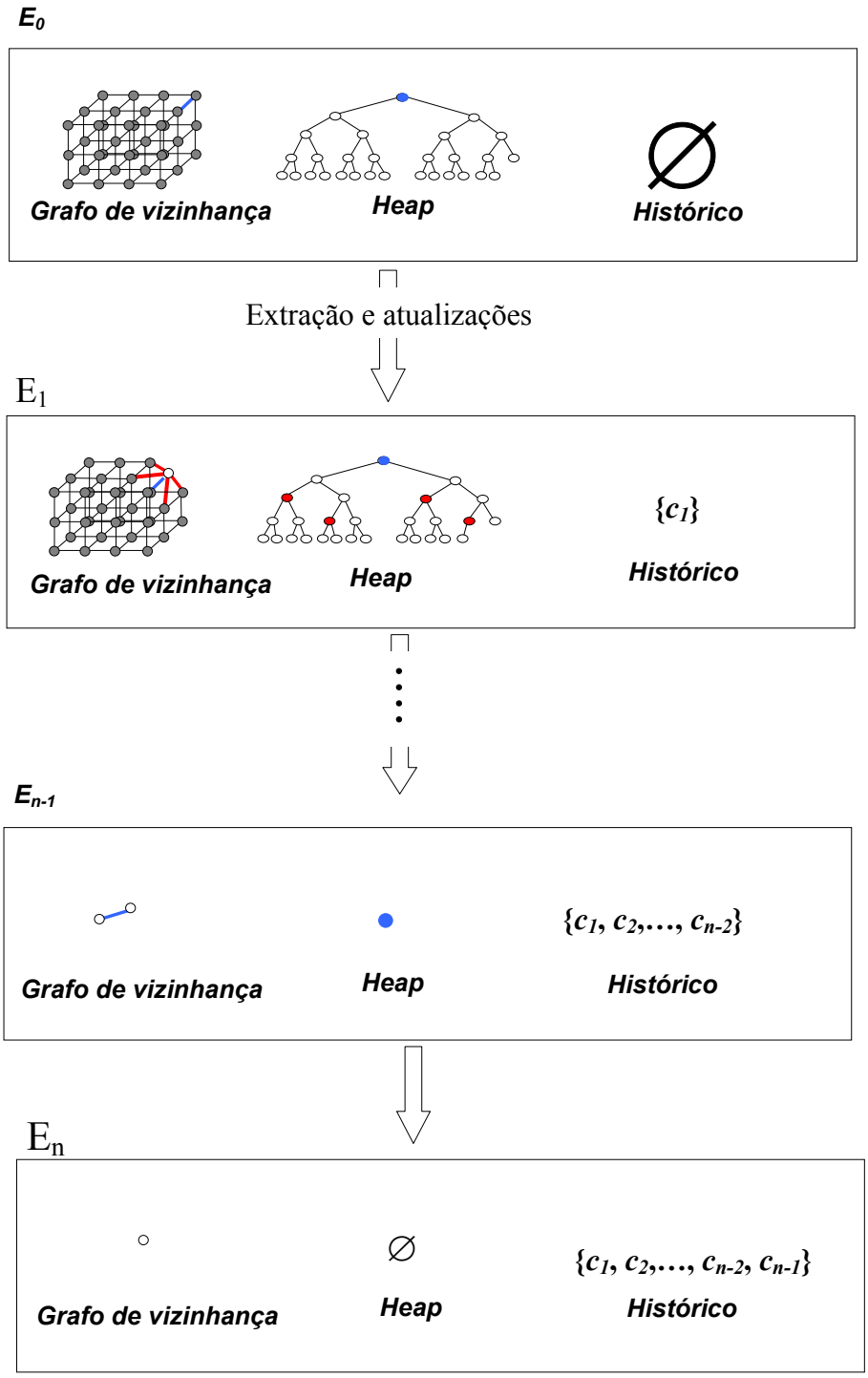


Figura 4.6 - Algoritmo incremental. O algoritmo inicia a partir de E_0 e, a cada passo, extrai do *heap* e efetua o colapso de menor custo (em azul) criando uma nova célula (em branco) no grafo de adjacência. Além disso, atualiza a estrela do novo nó (em vermelho).

4.1.3 Hierarquia

O histórico gerado pelo algoritmo incremental apenas codifica de forma seqüencial as operações de colapso, criando dependências cronológicas entre elas. Entretanto, através da análise de interferência das operações entre si, podem-se identificar operações independentes passíveis de serem efetuadas simultaneamente.

A exemplo da hierarquia de vértices [Hop97, Lue97, Xia96], pode-se construir em tempo O(***n'***), diretamente a partir do histórico de operações, uma estrutura que codifica a série de colapsos e suas dependências. A estrutura resultante é uma árvore binária (Figura 4.7) que, assim como a hierarquia de vértices, é apropriada para a extração de modelos adaptativos em tempo real. Cada nó da árvore corresponde a uma célula e dois arcos irmãos a uma operação de colapso.

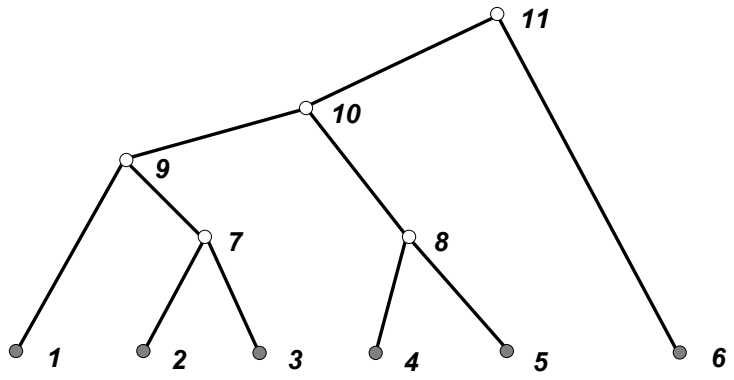


Figura 4.7 - Estrutura da hierarquia de células. As folhas correspondem às células do modelo original. Os nós internos correspondem às macro-células.

Na verdade, a hierarquia pode ser construída durante a execução do algoritmo incremental sem a necessidade de existir explicitamente um histórico. Para isto, é necessário gerenciar uma **floresta** de hierarquias de células. A floresta inicial é formada por tantas árvores quantas forem as células ativas do reservatório. Cada árvore, nesta etapa inicial, é formada por um único nó. Durante o laço de repetição, quando um colapso é efetuado, duas árvores são fundidas. Ao final do laço de repetição, quando não existirem mais colapsos, ***A*** será formado por apenas um nó e a floresta será composta por somente uma árvore (Figura 4.8).

O pseudo-código abaixo ilustra o algoritmo proposto considerando a hierarquia em vez do histórico:

*1° passo: constrói o espaço de colapsos eliminando células inativas //O(**n**)*
*2° passo: constrói o heap //O(**a**)*
*3° passo: constrói a floresta inicial //O(**n'**)*
4° passo: enquanto houver colapsos
{
*extrai o menor colapso //O(log**a**)*
*atualiza o grafo de vizinhança //O(**g**)*
*atualiza o heap //O(**g**log**a**)*
atualiza a floresta //O(1)
}

A complexidade final do algoritmo continua sendo O(***n'***log***n'***), apesar da alteração da complexidade do terceiro passo.

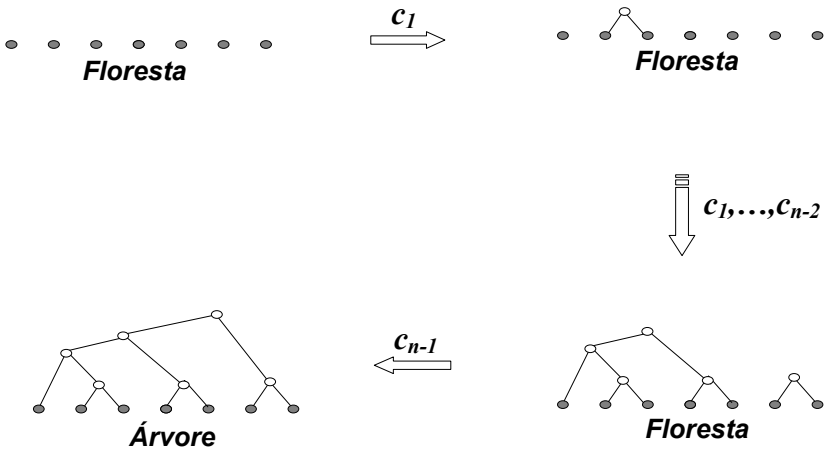


Figura 4.8 - Construção da hierarquia a cada iteração do algoritmo.

4.2 Características da Operação de Colapso

Tipicamente, a eficiência das operações incrementais é decisiva para o bom desempenho dos algoritmos gulosos de multi-resolução, pois elas realizam cálculos computacionalmente custosos [Gar99] que definem qual deve ser a nova geometria ótima que minimiza o erro segundo alguma norma pré-estabelecida. Como, na

maioria das vezes, a solução ótima é computacionalmente muito cara, heurísticas ou aproximações passam a ser utilizadas. Nessas heurísticas existe, geralmente, uma relação inversa entre tempo de processamento e qualidade do resultado obtido, ou seja, cálculos mais exatos, que produzem melhores resultados, são normalmente mais custosos.

4.2.1 Geometria e Valor de Propriedade

Visando alcançar uma boa eficiência computacional, a operação de colapso utiliza a vizinhança entre as células para calcular uma alternativa viável à macro-célula ótima. Isto é feito através da definição de três padrões fixos de colapso para o cálculo da macro-célula, um para cada direção ***I***, ***J*** e ***K*** da vizinhança entre células (Figura 4.9). Ou seja, se células são vizinhas em ***I***, por exemplo, é aplicado o padrão correspondente a ***I***. Os padrões são definidos através dos rótulos das faces (veja a Seção 3.2).

Para manter a estrutura de dados compacta, o colapso não gera novos vértices, aproveitando os vértices originais da malha de simulação. Portanto, a malha de multi-resolução tem o mesmo número de vértices da malha original. A Figura 4.9 ilustra o processo de formação da macro-célula nas 3 direções ***I***, ***J*** e ***K***. Neste processo, as faces extremas do intervalo ***I*** (ou ***J*** ou ***K***) são preservadas e as quatro novas faces da macro-célula são obtidas unindo-se as arestas correspondentes.

Neste momento, é importante ressaltar que há mais uma dimensão a ser levada em consideração pela operação de colapso: a propriedade das células definida em $\mathfrak{R}$. Assim, dada uma propriedade $\boldsymbol{i_k}$, o valor da propriedade da macro-célula, ***c,*** adotado neste trabalho, é a média ponderada dos valores de propriedade das células de entrada, ***a*** e ***b***, em relação ao seus volumes:

$$i_k(c) = \frac{i_k(a)\cdot v(a) + i_k(b)\cdot v(b)}{v(a) + v(b)}, \qquad \textbf{(4.1)}$$

onde $\boldsymbol{i_k(a)}$ e $\boldsymbol{i_k(b)}$ são os valores da ***k***-ésima propriedade nas células ***a*** e ***b*** e $\boldsymbol{v(a)}$ e $\boldsymbol{v(b)}$ são os volumes das células ***a*** e ***b***.

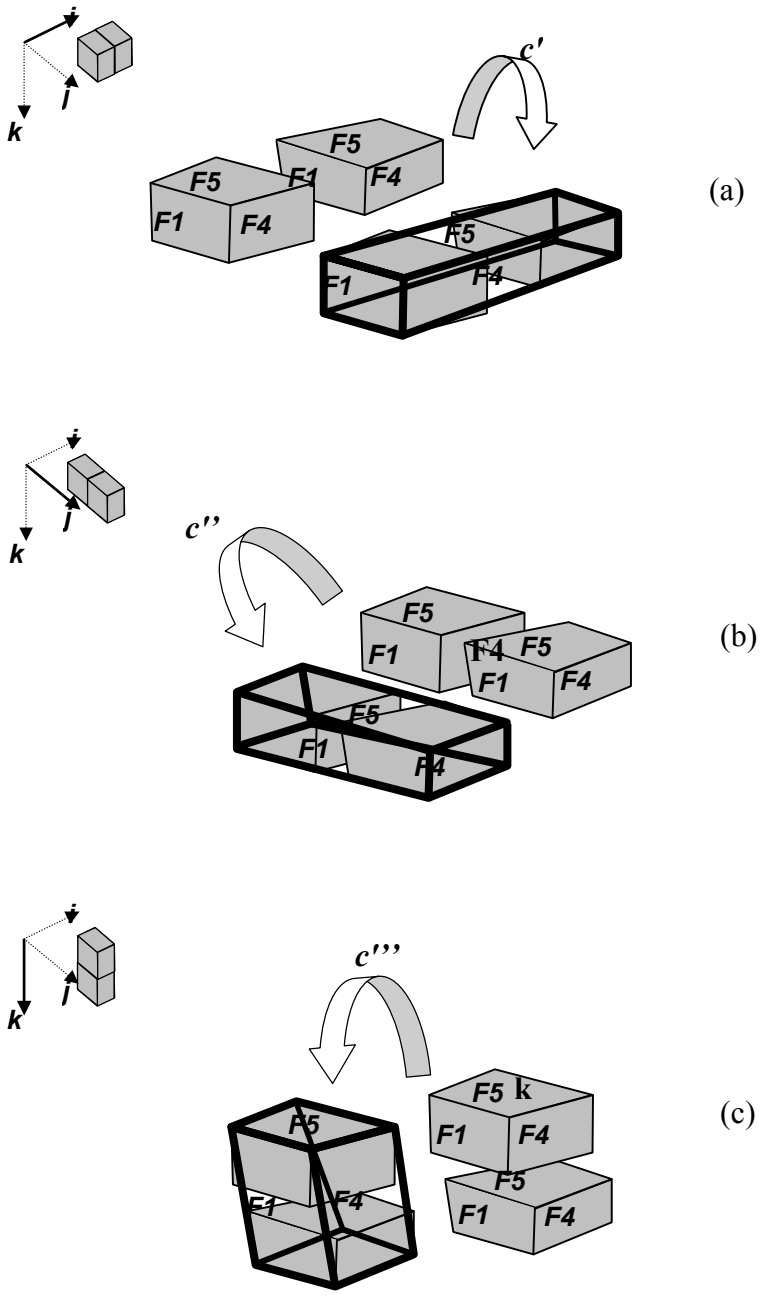


Figura 4.9 - Padrões para o cálculo da macro-célula baseado na vizinha do espaço paramétrico. Em (a), o padrão na direção *I*. Em (b), o padrão na direção *J*. Em (c), o padrão na direção *K*.

A equação (4.1) visa considerar características perceptuais no cálculo, uma vez que as células de maior volume possivelmente ocupem uma maior área no plano de projeção que forma a imagem.

4.2.2 Erros Geométrico e de Propriedade

Para cada macro-célula candidata a ser formada, é necessário calcular um custo levando-se em conta os erros geométrico e de propriedade. O erro pode ser computado através de alguma média ponderada entre os dois erros citados ou pela priorização de um erro em relação aos outros.

O erro geométrico entre duas células, ***a*** e ***b***, e uma macro-célula, ***c***, que as represente pode ser definido por uma norma semelhante à distância de Hausdorff [Pre85]. Ou seja, se ***S*** é o conjunto dos pontos da fronteira de ***a*** e ***b*** e ***T*** é o conjunto dos pontos da fronteira de ***c*,** então o erro, $\varepsilon_g(c)$, é definido como sendo:

$$\varepsilon_g(c)=\varepsilon(T,S)=\max(d(s,T),d(t,S)), \qquad \textbf{(4.2)}$$

onde ***d(s,T)*** é a distância máxima dos pontos da fronteira de ***S*** para a fronteira de ***T*** e ***d(t,S)*** é a distância máxima dos pontos da fronteira de ***T*** em relação à fronteira de ***S***. A escolha dos pontos de fronteira, e não do interior das células, baseia-se no interesse pela imagem gerada pela restituição (*rendering*) das faces do modelo. Ou seja, como somente as fronteiras das células são utilizadas na visualização, o algoritmo proposto se concentrou na diferença entre elas.

Para aumentar a eficiência do algoritmo proposto e tirar proveito das características especiais do modelo de RNPs, optou-se por aproximar ***d(s,T)*** e ***d(t,S)*** por duas distâncias fáceis de calcular. A primeira é estimada como sendo a maior distância dos vértices de ***S*** até as faces de ***T*** e a segunda como sendo a distância entre os vértices de ***S*** que foram eliminados na formação da macro-célula. A Figura 4.10 ilustra estas distâncias com setas em preto e azul, respectivamente. A justificativa para aproximar ***d(t,S)*** por esta distância vem do fato de que, quanto maior for o espaço entre as células (*gap*), maior será a distância dos pontos de ***T*** ao conjunto ***S***.

O erro de propriedade da macro-célula na propriedade i_k é dado pela função $\varepsilon i_k(T, S)$, que na sua forma reduzida é:

$$\varepsilon i_k(c) = \max\{|\, i_k(a) - i_k(c)\,|, |\, i_k(b) - i_k(c)\,|\}, \qquad \textbf{(4.3)}$$

Para considerar ***L*** propriedades no cálculo do erro pode-se utilizar a equação:

$$\varepsilon i(c)=\frac{1}{L}\sum_{l=0}^{L}\max\{|il(a)-il(c)|,|il(b)-il(c)|\}, \quad \textbf{(4.4)}$$

Tanto o erro geométrico quanto o erro de propriedade respeitam à norma $\boldsymbol{L_\infty}$ localmente e é importante observar que o valor de propriedade associado à macro-célula na equação (4.1) não minimiza a norma $\boldsymbol{L_\infty}$ utilizada na equação (4.3).

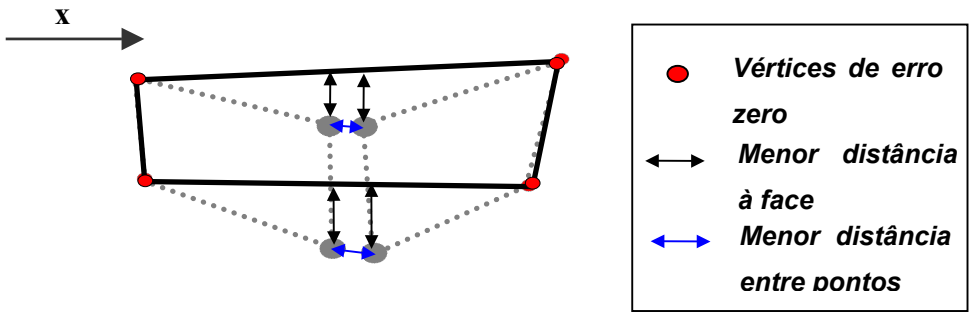


Figura 4.10 - Distâncias consideradas no cálculo do erro geométrico.

4.2.3 Tratamento de Múltiplas Propriedades

Uma vez que o modelo de um reservatório comumente possui dezenas de propriedades diferentes associadas, é inviável construir uma estrutura de MR para cada uma delas. Uma alternativa seria utilizar a equação (4.4) no cálculo de erro e considerar todas as propriedades numa única estrutura. Entretanto, além de aumentar o tempo gasto para a construção da hierarquia, esta solução não garante bons resultados.

Neste trabalho, optou-se por considerar apenas o erro geométrico na construção da estrutura MR. Os cálculos referentes às propriedades são feitos por demanda, de acordo com a necessidade de visualização. Percebe-se que estes cálculos não mais afetarão a estrutura da hierarquia, apenas adicionarão novos valores de propriedade e erro aos seus nós.

4.2.4 Propriedade de Ordenação Parcial

Os cálculos do erro geométrico e de propriedade utilizados pela operação de colapso garantem limites de erro segundo a norma L_∞ apenas entre aproximações do reservatório geradas por cortes na hierarquia que possuam, no máximo, um nível de diferença.

Analisando a versão unidimensional do problema (hierarquia de arestas para linhas poligonais), pode-se perceber as implicações deste fato (Figura 4.11). O erro geométrico $\varepsilon_g(l_7)$ é o erro real entre $\{l_7\}$ e $\{l_5, l_6\}$. Entretanto, na ilustração, fica evidente que o erro geométrico real entre $\{l_7\}$ e as folhas $\{l_1, l_2, l_3, l_4\}$ é maior do que $\varepsilon_g(l_7)$. Ou seja, a estrutura de multi-resolução exemplificada é incapaz de garantir um limite de erro em relação ao modelo original.

Isto seria exatamente o que aconteceria com a hierarquia de células se fossem utilizados diretamente os erros da operação de colapso. Refazer os cálculos da operação de forma que eles sejam realizados diretamente sobre o modelo original é demasiadamente custoso e violaria a característica de visualização que requer baixo tempo de pré-processamento (veja a Seção 3.3). Alternativamente, poderia-se pensar em utilizar para o erro de uma macro-célula o maior valor dentre o erro calculado pela operação de colapso e a soma dos erros das células filhas. Assim, sendo a e b as células de entrada de um colapso e c a macro-célula gerada, tem-se:

$$\varepsilon'_g(c)=\max\{\varepsilon_g(c),\varepsilon_g(a)+\varepsilon_g(b)\}.$$

Provavelmente, este novo erro seria um limite superior para o erro real. Entretanto, o fato desse valor ser utilizado pelo *heap* para avaliar o custo de um colapso durante a construção da estrutura de MR torna essa super-estimação inadequada, pois colapsos de qualidades distintas poderiam ser classificados (estimados) da mesma forma.

A solução utilizada neste trabalho consiste em considerar o erro como o máximo entre o erro da operação de colapso e os erros das células de entrada da operação:

$$\varepsilon'_g(c)=\max\{\varepsilon_g(c),\varepsilon_g(a),\varepsilon_g(b)\}, \qquad \textbf{(4.5)}$$

Esta solução busca uma medição mais realista do erro sem, com isto, aumentar o custo dos cálculos, mas não garante um limite de erro em relação ao modelo original. Espera-se que, no caso de RNPs, a boa coplanaridade entre as células dos reservatórios e o caráter volumétrico da operação de colapso façam com que os resultados sejam satisfatórios.

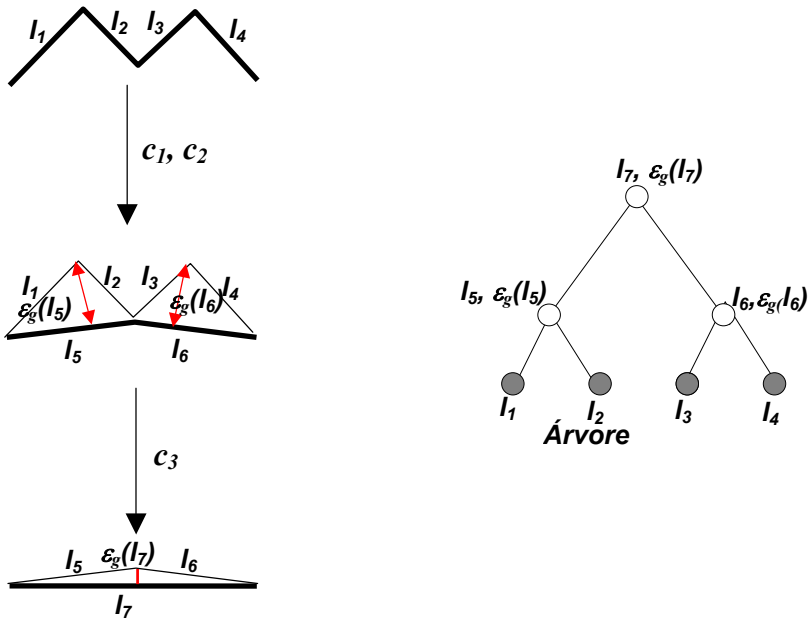


Figura 4.11 - Exemplo unidimensional: colapsos de arestas. O erro, $\varepsilon_g(I_7)$, associado ao colapso c_5 é menor do que os erros, $\varepsilon_g(I_5)$ e $\varepsilon_g(I_6)$, associados aos colapsos c_1 e c_2.

A utilização da equação (4.5) faz com que a estrutura de MR tenha uma propriedade de ordenação parcial decrescente em relação ao erro desejável para estruturas de multi-resolução.

Sejam ***a*** e ***b*** as células de entrada de um colapso e ***c*** a macro-célula gerada, tem-se que:

$$\varepsilon'_g(c) >= \varepsilon'_g(a) \quad \wedge \quad \varepsilon'_g(c) >= \varepsilon'_g(b).$$

E, portanto:

> "*o erro associado a um nó interno da hierarquia é sempre maior ou igual aos erros associados aos seus nós filhos*", **P.4.2**

O mesmo raciocínio pode ser aplicado para o caso do erro de propriedade. Assim, tem-se a equação:

$$\varepsilon i'_I(c)=\max\{\varepsilon i_I(c),\varepsilon i_I(a),\varepsilon i_I(b)\}, \qquad \textbf{(4.6)}$$

Com isso:

$$\varepsilon i'_I(c)>=\varepsilon i'_I(a) \quad \wedge \quad \varepsilon i'_I(c)>=\varepsilon i'_I(b).$$

E, portanto, levando em consideração uma propriedade específica:

"*o erro de propriedade associado a um nó interno da hierarquia é sempre maior ou igual aos erros de propriedade associados aos seus nós filhos*". ***P.4.3***

4.3 Propriedades de Extração

Em relação à estrutura combinatorial utilizada pelo MMR proposto, é válido fazer algumas observações importantes. Em primeiro lugar, nota-se que qualquer corte na hierarquia de MR gerado por um caminhamento pré-fixado corresponde a uma aproximação do reservatório que abrange todo o seu domínio sem que haja porções do domínio cobertas por mais de uma célula (Figura 4.12). Uma vez que são permitidos modelos não-conformes, o fato anterior constitui a regra básica de extração de modelos do MMR proposto.

Considerando isso, pode-se atribuir à hierarquia de células algumas propriedades que são bastante desejáveis a estruturas de MR:

- alto poder de expressão – o número possível de aproximações corresponde ao número possível de cortes, que é significativo uma vez que a estrutura é binária, permitindo uma grande variação de resolução ao longo do domínio do reservatório;

- extração ótima proporcional ao tamanho da aproximação – cada corte na árvore está associado a uma sub-árvore, como ilustrado na Figura 4.13. Somente os nós da sub-árvore são visitados pelo caminhamento que gera a aproximação. Uma vez que o número de nós de uma árvore binária é aproximadamente o dobro do seu número de folhas, tem-se que a complexidade de extração é linear em relação ao número de células da aproximação.

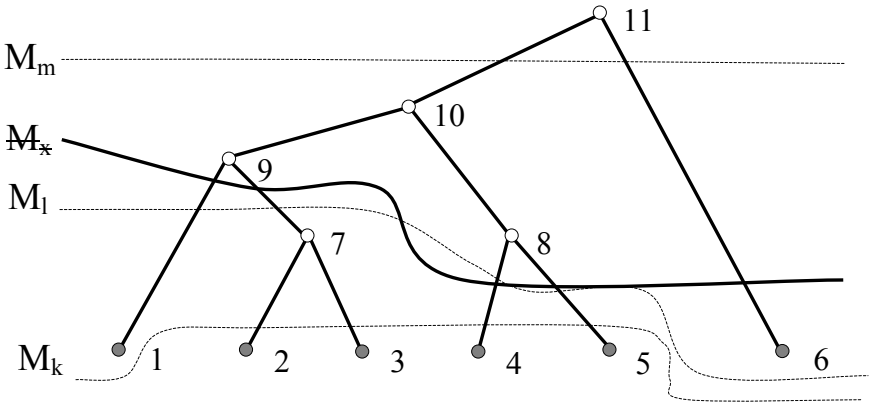


Figura 4.12 – Corte na árvore. O modelo gerado por um corte corresponde às folhas da sub-árvore que está acima do corte. Os cortes M_k, M_l e M_m formam três modelos válidos que abrangem todo o domínio do objeto. O corte inválido ~~M_x~~ apresenta redundância e não pertence ao conjunto dos cortes gerados pelo caminhamento pré-fixado.

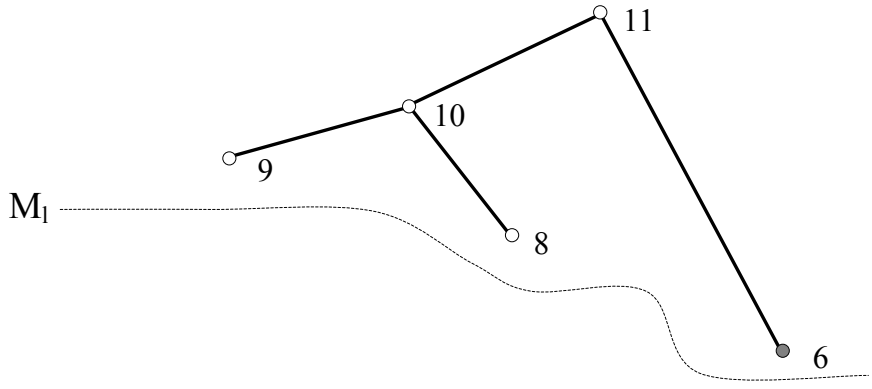


Figura 4.13 – Sub-árvore associada ao corte M_l. A árvore é a mesma da Figura 4.12.

5 Extração de Modelos

O objetivo deste capítulo é descrever os algoritmos de extração de malhas a partir da hierarquia de células hexaédricas descrita no Capítulo 4. Considerando a regra de extração básica, o objetivo é criar algoritmos que extraiam, em tempo de exibição, malhas em diferentes resoluções, dependentes: (a) do erro da aproximação (*error-dependent*), (b) da posição da câmera (*view-dependent*) e (c) do número de células da aproximação (*polygon-budget*).

5.1 Extração Dependente do Erro Geométrico

A extração **dependente do erro geométrico** objetiva achar $\boldsymbol{m} \in \mu$', de menor complexidade, cuja diferença para o modelo original seja menor do que (ou igual a) um erro $\varepsilon \in \Re^+$ pré-definido. Neste tipo de extração, o erro ε é o único parâmetro do contexto de visualização, ou seja, o contexto é definido diretamente no espaço do objeto, não importando, por exemplo, quais são os parâmetros de câmera.

O algoritmo de extração consiste em percorrer a árvore de cima para baixo e, a cada nó da hierarquia, verificar se o erro a ele atribuído é menor do que ε (ou igual a ele). Se for menor ou igual, a célula associada ao nó é incluída na lista de células visíveis; caso contrário, o algoritmo prossegue recursivamente com os filhos do nó. Caso o nó

seja uma folha, ele será necessariamente incluído na lista, pois possui erro zero. No final, a lista de células visíveis corresponde ao corte na árvore que representa um modelo ***m***. Este algoritmo pode ser ilustrado pelo pseudo-código a seguir:

```
struct estr_célula;
typedef estr_célula tipo_célula;

struct estr_nó_hier {
                    est_nó* esquerdo; // filho esquerdo
                    est_nó* direito;      // filho direito
                    float erro;
                    tipo_célula* célula;
                    tipo_aabb* aabb; //caixa envolvente da região
};
define esrt_nó_hier  tipo_nó;

void Extração_dependente_do_erro (float ε, tipo_nó* nó) {
        if ((nó→ÉFolha()) || (nó→erro ≤ ε))
                Seleciona  (nó.célula);
        else
        {       Extração_dependente_do_erro (ε, nó→esquerdo);
                Extração_dependente_do_erro (ε, nó→direito );
        }
}
```

O procedimento *Extração_dependente_do_erro* deve receber a raiz da hierarquia no início da recursão.

Devido à propriedade de ordenação parcial da estrutura de MR (P.4.2), pode-se provar que ***m***, restrito ao espaço dos modelos μ', é ótimo. A prova é baseada em dois fatos contraditórios:

1. a expressão condicional efetuada pelo algoritmo garante que todos os nós acima do corte gerado possuem erro maior do que ε;
2. se um corte não é o de menor complexidade, existe pelo menos um par de nós irmãos que pode ser substituído pelo seu nó pai sem que o erro seja violado, ou seja, o erro associado ao pai é menor do que ε.

Consideremos o corte M_i gerado pelo algoritmo de extração parametrizado pelo erro ε. S é o conjunto formado por todos os nós acima de M_i. Do fato 1, sabe-se que todo nó p pertencente a S possui necessariamente seu erro maior do que ε. Ou seja:

$$\forall p \in S \rightarrow p.erro > \varepsilon.$$

Supondo que M_i não gere o modelo de menor complexidade, a partir do fato 2, espera-se que exista p' pertencente a S tal que o erro associado a p' seja menor do que (ou igual a) ε. Ou seja:

$$p' \in S \ \wedge \ p'.erro \leq \ \varepsilon.$$

Isso mostra que a suposição é inválida e que, portanto, se M_i é gerado pelo algoritmo de extração, então M_i é o modelo de menor complexidade que respeita o erro ε.

5.2 Extração Dependente da Câmera

A extração **dependente da câmera** é própria para aproveitar as distorções causadas por uma transformação projetiva, que valoriza mais as regiões próximas à câmera em relação às regiões distantes (Figura 5.1).

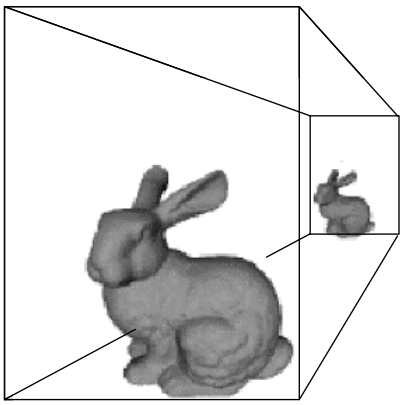


Figura 5.1 - Deformação da projeção perspectiva.

De forma geral, as regiões mais próximas necessitam ser desenhadas numa resolução mais alta do que as regiões mais distantes. Por outro lado, durante uma animação, à medida que a câmera se aproxima de uma região, causando o aumento da sua resolução, outras regiões passam a ficar fora de seu volume de visão, podendo ser desconsideradas para a visualização. Em alguns casos, os dois efeitos tendem a compensar um ao outro no sentido de que, idealmente, o número de polígonos desenhados mantém-se num valor médio constante.

São necessários dois tipos de testes para os algoritmos de busca dependentes da câmera: um para descartar os objetos que estão fora do volume de visão e outro para decidir qual a variação da resolução ao longo do domínio.

5.2.1 Descarte

O descarte é responsável pela eliminação de regiões que estão fora do volume de visão. Para realizar o descarte de forma eficiente, é necessário descartar vários polígonos ao mesmo tempo. Estruturas hierárquicas que formam conjuntos disjuntos de *clusters* são reconhecidamente bem utilizadas na realização desta tarefa [Gar98, DeRose98].

Uma vez que a estrutura de MR utilizada é uma hierarquia, a propriedade **P.4.1** garante que essa estrutura pode ser vista como uma estrutura de *clusters* disjuntos de células, na qual cada colapso (ou nó) representa a união entre dois *clusters*. Se associarmos a cada nó da hierarquia de células o volume envolvente de seu *cluster*, temos uma hierarquia de volumes envolventes que pode ser usada para realizar o descarte.

Os cálculos do volume envolvente de cada *cluster* devem ser feitos juntamente com a construção da hierarquia. Sendo assim, eles influenciam o tempo total gasto em pré-processamento, o qual, como dito anteriormente, objetivamos reduzir. Para realizá-los eficientemente, o volume envolvente utilizado neste trabalho foi a caixa alinhada com os eixos, *aabb* (*aligned-axis bounding box*), pois podemos achar as caixas envolventes mínimas em tempo linear no número de nós da hierarquia.

Além de permitir uma rápida construção, a utilização da *aabb* possui um teste de pertinência (*inside-outside test* [Tom99]) eficiente e invariável a uma possível deformação em $\boldsymbol{Z}$ na geometria do reservatório prevista pelas **características de visualização** do problema, ou seja, uma *aabb* continua sendo uma *aabb* após uma deformação em $\boldsymbol{Z}$.

5.2.2 Cálculo do Erro Projetado

Para decidir a resolução adequada de uma região do reservatório adaptando-a em relação à câmera, foi utilizado um cálculo baseado na projeção, no plano de imagem, do erro geométrico local existente na região (Figura 5.2).

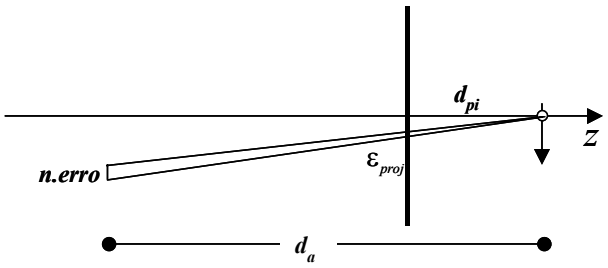


Figura 5.2 – Redução do erro em função da profundidade.

Dado um nó ***n*** da hierarquia e a célula ***c*** associada a ele, o erro projetado relativo a ***dom***(***c***) é dado pela equação:

$$\varepsilon_{proj} = n.erro \times (d_{pi}/d_a),$$

onde d_{pi} é a distância do centro do plano da imagem à câmera e d_a é a menor distância da caixa envolvente da célula à câmera.

O cálculo projetado é particularmente bom porque reproduz o comportamento da projeção cônica utilizada pela câmera (considerando uma câmera em perspectiva). Além disso, ele pode ser facilmente estendido para levar em consideração a resolução da imagem gerada. Se a imagem possui dimensões ***lx*** e ***ly***, no sistema global, e dimensões ***w*** e ***h***, no sistema de janela, então o erro projetado pode ser mapeado para o erro em *pixels* através da equação:

$$\varepsilon_{proj} = (n.erro \times (d_{pi}/d_a)) \times (max(w, h)/max(lx, ly). \qquad \textbf{(5.1)}$$

5.2.3 Algoritmo de Extração

O algoritmo de extração dependente da câmera é semelhante ao algoritmo de extração dependente do erro geométrico. A única diferença é a substituição do erro geométrico pelo erro projetado e o teste de descarte. Assim, o pseudo-código do algoritmo é:

```
estrutura estr_parâmetros_de_visualização {
                    float x, y;
                    float lx, ly;
                    float d_pi, d_a;
                    float ε;
};
define estr_parâmetros_de_visualização tipo_params_vis;

float Projeta (tipo_params_vis v, float erro) {
                    retorna (erro/(v.d_pi - v.d_a)) × (max(v.x, v.y)/ max(v.lx, v.ly));
}

void Extração_dependente_da_câmera (tipo_params_cam v, tipo_nó* nó) {
          if Descarta(nó→aabb)
                    return;
          else if ((nó→Éfolha()) || (Projeta(nó→erro) ≤ v.ε))
                    Seleciona  (nó.célula);
          else {
                    Extração_dependente_da_câmera (v, nó→esquerdo);
                    Extração_dependente_da_câmera (v, nó→direito);
          }
}
```

O procedimento *Extração_dependente_da_câmera* também deve receber a raiz da hierarquia no início da recursão. O procedimento *Descarta* é responsável por realizar o teste de pertinência do volume envolvente da região do reservatório contra o volume de visão da câmera. Maiores detalhes a respeito desse teste podem ser encontrados nos trabalhos de Thomas [Tom99] e Hofmam [Hof02].

A prova de que os modelos extraídos por este algoritmo são ótimos é idêntica à prova do algoritmo de extração dependente do erro geométrico, desde que a propriedade de ordenação parcial da hierarquia seja válida também para o erro projetado. Este fato é verdadeiro se considerarmos que a distância da *aabb* de um nó pai à câmera é sempre menor do que (ou igual a) a distância da *aabb* de seus filhos à câmera (veja equação 5.1).

5.3 Extração Dependente do Número de Polígonos

O algoritmo de extração **dependente do número de polígonos** é adequado à natureza interativa da visualização, pois possibilita o controle preciso de um dos fatores

determinantes para a interação: o número de polígonos enviados para o canal de visualização (*rendering pipeline*).

Dado um número de células ***NP***, o objetivo é achar, dentre todos os modelos de complexidade igual a ***NP*** e pertencentes a μ', aquele que possui o menor erro geométrico (ou projetado).

O algoritmo utilizado é guloso. Ele inicia com o corte mínimo da hierarquia, ou seja, com o corte cujo modelo associado é formado apenas pelo nó raiz e, a cada passo, altera o corte atual descendo localmente pelo nó que possui maior erro. A etapa de descida é repetida até que o corte atual possua ***NP*** nós ou que seja formado apenas por folhas.

Considerando que o número de polígonos pré-estabelecido deve ser limitado pelo número de polígonos do modelo original, o pseudo-código do algoritmo, em alto nível, é:

```
void Extração_dependente_do_número_de_polígonos (int NP, tipo_nó* nó) {
        tipo_nó_lista* corte;
        corte→elem = nó;
        int tamanho = 1;

        while (tamanho < NP) {
                Desce(corte);
                Tamanho++;
        }
        Seleciona (corte);
}
```

O método *Desce* recebe como parâmetro a lista que representa o corte atual e é responsável por retirar o nó de maior erro e inserir seus nós filhos. Aqui, vale uma observação quanto à implementação: deve ser utilizado um *heap* para controlar eficientemente a escolha do maior erro do corte. Com esse *heap*, pode-se calcular o erro máximo do novo corte em O(log ***NP***). Assim, a extração do modelo pode ser feita em O(***NP*** log ***NP***), perdendo a propriedade de extração linear, mas ainda permitindo a obtenção de taxas de iteração adequadas na prática.

Indutivamente, pode-se provar que os modelos gerados pelo algoritmo são ótimos. Para tanto, é importante considerar a propriedade P.4.2 e, também, que o erro associado a um corte é o máximo dentre os erros de todos os nós que formam o corte. Para ***NP*** igual a 1 e 2, é trivial verificar que os modelos são ótimos, pois, no espaço de modelos, só há um modelo composto por uma célula e um modelo composto por duas células. Para ***NP*** igual a 3, também é fácil verificar que o algoritmo guloso funciona.

Supondo que o algoritmo seja ótimo para ***NP*** igual a ***L***, pode-se verificar que a troca efetuada pela função *Desce* gera também o corte ótimo para ***L***+1, pois essa é a única troca que diminui o erro existente no corte anterior. Isso mostra que a aplicação seqüencial das trocas efetuadas por *Desce*, a começar por ***NP*** igual a 3, leva somente a modelos ótimos.

5.4 Dependência do Erro de Propriedade

Como visto no capítulo anterior, uma propriedade pode ser adicionada à estrutura de MR para ser visualizada. Portanto, é interessante que os algoritmos de extração de modelos possibilitem considerar o erro de propriedade na geração das aproximações.

Essa característica pode ser facilmente introduzida. Neste trabalho, os algoritmos dependentes do erro geométrico e da câmera incorporam-na através da efetuação de um teste duplo, permitindo que um nó seja selecionado se tanto o limite de erro geométrico quanto o de propriedade forem satisfeitos. No caso do algoritmo dependente da câmera, o erro de propriedade pode ser invariante à projeção.

No algoritmo de extração dependente do número de polígonos, a solução adotada foi um pouco diferente. Em vez de incorporar o erro de propriedade na função de avaliação de custo do *heap*, foi feita uma mescla entre as dependências: o algoritmo continua essencialmente dependente do número de polígonos, porém um limite máximo de erro de propriedade é permitido. Se não for possível respeitar esse limite, o algoritmo acaba retornando uma malha com mais polígonos do que o desejado.

6 Resultados

Os Capítulos 2 e 3 descreveram o MMR proposto para as malhas de simulação em RNPs, analisando suas principais características. A análise de complexidade foi feita tanto para o algoritmo de construção da estrutura de MR quanto para os algoritmos de extração de malhas. Entretanto, essa análise não é suficiente para garantir que os tempos reais de processamento sejam satisfatórios para a aplicação nem está relacionada à qualidade das aproximações geradas. Além disso, é necessário verificar se as características de visualização do problema aqui tratado foram atendidas.

Dessa forma, este capítulo tem por objetivos: (a) analisar o desempenho empírico do algoritmo de pré-processamento, (b) analisar a qualidade das aproximações geradas pelo MMR, (c) analisar as taxas de interatividade alcançadas sob diversas condições e (d) verificar se os requisitos de visualização foram satisfeitos.

Para facilitar o entendimento das análises efetuadas, o capítulo é dividido em três etapas distintas: *pré-processamento*, *extração de malhas* e *características de visualização*.

6.1 Considerações Iniciais

A implementação do MMR proposto, com a qual os testes deste trabalho foram realizados, foi feita nas linguagens C e C++ e desenvolvida com a ferramenta Visual C++ 6.0 da Microsoft Corporation. Apesar do ambiente de desenvolvimento estar relacionado ao Sistema Operacional (S. O.) Windows, o código é portável para as plataformas Linux e Unix. A interface de aplicação gráfica OpenGL [OpenGL99] foi utilizada para aproveitar possíveis recursos de alto desempenho disponibilizados por placas gráficas.

Os testes foram efetuados em quatro plataformas (plt.) de *hardware* distintas operadas pelo Microsoft Windows 2000, formando as quatro configurações descritas na Tabela 1.

Tabela 1 - Plataformas de *hardware*.

	Processador	**Freqüência (MHz)**	**Memória Principal (MB)**	**Processador Gráfico**	**Memória de Vídeo (MB)**
Plataforma I	Intel Pentium III	850	256	---------	---------
Plataforma II	Intel Pentium IV	2.000	512 (RAM BUS)	---------	---------
Plataforma III	Intel Pentium III	850	256	Rage 128	32
Plataforma IV	Intel Pentium IV	2.000	512 (RAM BUS)	G-Force III	64

As malhas de simulação utilizadas para a bateria de testes estão descritas na Tabela 2, onde várias características foram relacionadas. O item "número de triângulos" (**NT**) foi inserido para proporcionar uma relação mais familiar com os tempos de renderização das malhas através de placas gráficas. Esse número é proporcional ao número de células ativas (**NC**):

$$NT = 12 \times NC.$$

A nomenclatura dada para as malhas obedece a ordem crescente em relação a NC e é seguida das terminações “bc” ou “cp” para indicar se a malha é *block-centered* ou *corner-point*, respectivamente. A única exceção é a I_bc, que recebe esta denominação por possuir uma geometria bastante irregular (Figura 6.1), nada típica às malhas de simulação, sendo utilizada para testar o MMR em um caso desfavorável. Um fato importante é que as malhas A_bc e D_cp são discretizações de uma mesma geometria, ou seja, representam dois casos de estudo de um mesmo reservatório.

Tabela 2 - Dados relativos às malhas de simulação.

	A_bc	**B_cp**	**I_bc**	**D_cp**	**E_bc**	**F_cp**
Tipo	block	corner	Block	corner	block	corner
MAX_I	40	42	120	83	95	175
MAX_J	34	30	100	45	32	154
MAX_k	5	11	8	23	126	12
N° de Células Ativas (NC)	3.703	9.033	11.850	31.550	42.065	50.964
N° de Células Inativas (NI)	3.097	4.827	84.150	54.335	340.975	272.436
N° de Triângulos (NT)	44.436	108.396	142.200	378.600	504.780	611.568
N° de Colapsos (NCOL)	10.002	25.408	29.845	90.026	117.495	135.965
N° de Prop. Iniciais	26	47	53	49	52	71
N° de Prop. Recorrentes	7	4	5	6	5	6
N° de Intervalos de Tempo	126	2	10	10	5	27

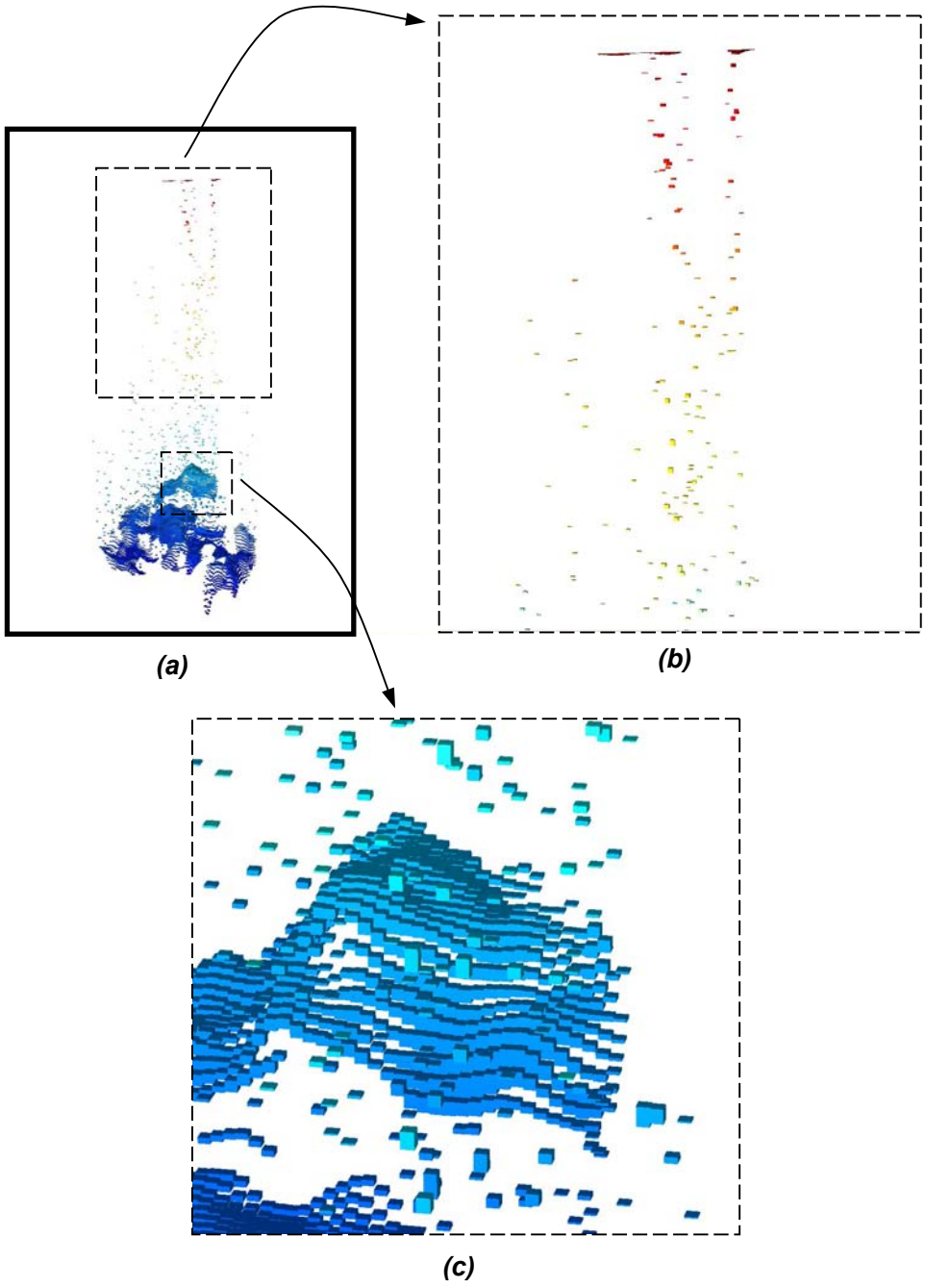


Figura 6.1 – Malha l_bc. Em (a), vista geral da malha l_bc. Em (b), região com comportamento bastante irregular e na qual praticamente não há faces em contato. Em (c), região melhor comportada, mas onde ainda percebe-se muita irregularidade.

6.2 Pré-processamento

Os testes referentes à etapa de pré-processamento visam analisar o tempo de construção da estrutura de multi-resolução bem como a quantidade de memória despendida nesta etapa. Para evitar efeitos indesejados na medição do tempo, apenas o tempo de **CPU** (Unidade Central de Processamento) consumido pelo processo foi medido. As plataformas de *hardware* utilizadas foram as plataformas I e II, observando que a quantidade de memória principal existente é suficiente para evitar a ocorrência de transferência de dados (*swap*) entre a memória principal e a secundária.

6.2.1 Análise do Tempo de Pré-processamento

Devido à característica do algoritmo, a aquisição dos tempos é feita em duas fases.

1ª Fase

A fase inicial é formada pelos três primeiros passos do algoritmo descrito na seção 4.1.3 e relacionados a seguir:

1° passo: constrói o espaço de colapsos eliminando células inativas
2° passo: constrói o heap
3° passo: constrói a floresta inicial

O primeiro passo é linear no número total de células e não apenas no número de células ativas, pois há a necessidade de percorrer as células inativas para desconsiderá-las. O segundo depende diretamente do número de colapsos e o terceiro depende do número de células ativas.

2ª Fase

A segunda fase é formada pelo laço de repetição principal, que é O(***n*** log ***n***), onde ***n*** é o número de células ativas. Nessa fase, são realizadas ***n*** iterações e cada uma delas possui complexidade O(***g*** log ***n***), onde ***g*** é o grau do nó escolhido. É importante observar que a variável ***g*** está sendo limitada pela constante ***G***, à qual atribuiu-se o valor 15 nesta implementação (ver seção 4.1.2).

Resultados

A Tabela 3 descreve os resultados obtidos e o Gráfico 1 e o Gráfico 2 permitem uma análise mais detalhada desses resultados. Nos gráficos, a curva "Inicial" representa os tempos da primeira fase do algoritmo adquiridos ao final do processamento de cada caso de teste em função do número de células ativas. O comportamento oscilatório deve-se às diferentes dependências atribuídas aos passos do algoritmo. Um bom exemplo são os comportamentos nas malhas E_bc e F_cp: apesar da F_cp possuir cerca de 8.000 células ativas e 17.000 colapsos a mais, as aproximadamente 70.000 células inativas a menos fizeram com que os tempos fossem próximos: 1,57 segundos para a E_bc e 1,62 segundos para a F_cp, ambos na plataforma I.

As demais curvas permitem averiguar o comportamento no tempo da segunda fase do algoritmo. Cada curva representa um caso de teste e indica o comportamento em função do número de células ativas existentes em cada iteração. Apesar do decrescimento constante do número de células ativas, as curvas apresentam um comportamento bem próximo do linear.

Em algumas curvas, em especial a E_bc, nas iterações finais, quando o número de células se aproxima de zero, pode-se perceber a influência da variável ***g*** nos tempos adquiridos. O aumento desses tempos nessa região da curva acontece devido à existência de iterações nas quais todos os colapsos remanescentes são proibidos e, assim, necessariamente nós de grau maior do que a constante ***G*** são selecionados.

Tabela 3 – Tempos de execução do algoritmo de pré-processamento anotados nas plataformas de *hardware* I e II, em segundos.

	Tempos de execução (s)					
	1ª fase		**2ª fase**		**Total**	
	Plt. I	**Plt. II**	**Plt. I**	**Plt. II**	**Plt. I**	**Plt. II**
A_bc	0,08	*0,06*	0,30	*0,16*	**0,38**	***0,22***
B_cp	0,25	*0,14*	0,79	*0,46*	**1,041**	***0,6***
I_bc	0,39	*0,21*	1,253	*0,511*	**1,292**	***0,721***
D_cp	0,911	*0,48*	3,465	*2,063*	**4,376**	***2,543***
E_bc	1,573	*0,912*	4,676	*2,834*	**6,249**	***3,746***
F_cp	1,623	*0,891*	4,456	*2,659*	**6,079**	***3,55***

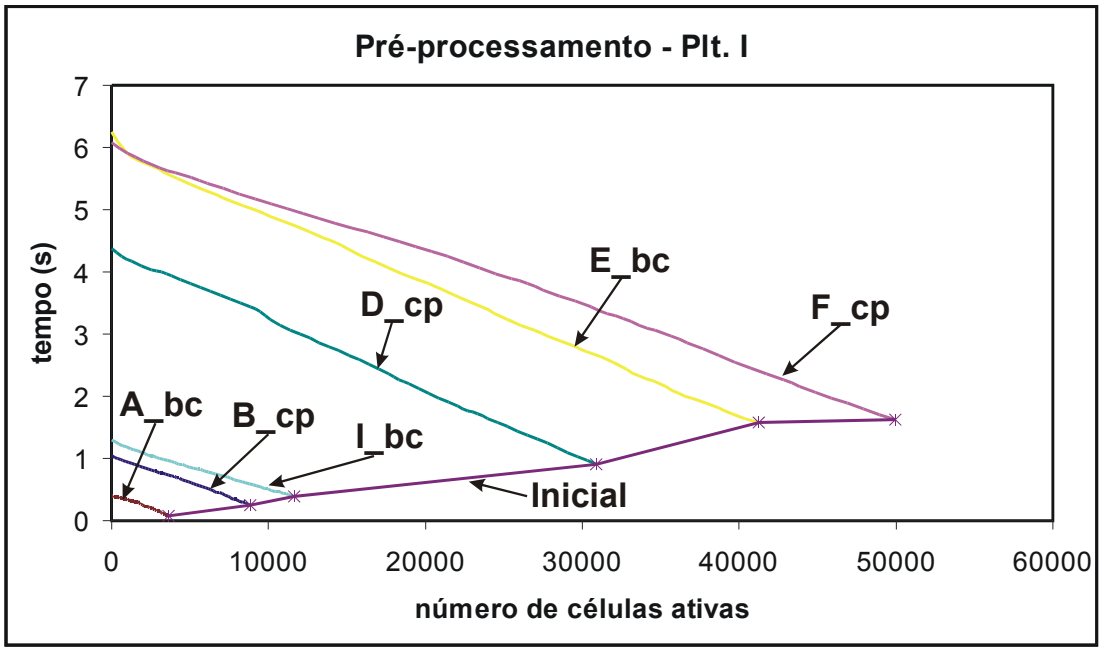


Gráfico 1 – Pré-processamento (plat. I). Resultados da fase de pré-processamento adquiridos na plataforma I. A curva "Inicial" representa as tomadas de tempo da primeira fase do algoritmo. As demais curvas representam, para cada caso de teste, o comportamento da segunda fase do algoritmo.

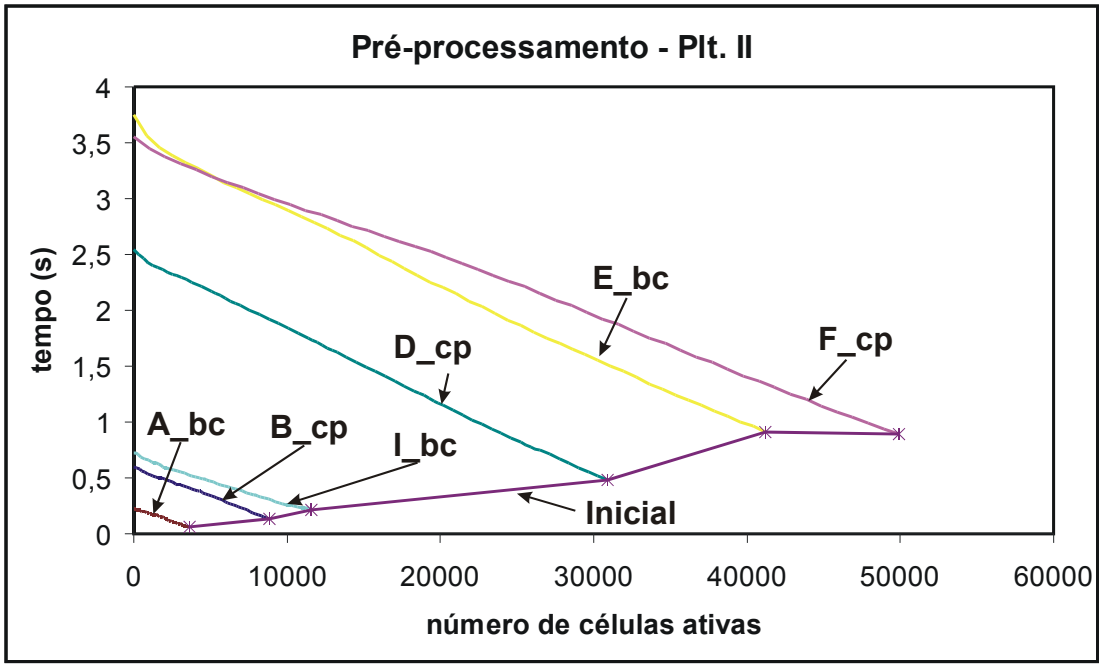


Gráfico 2 – Pré-processamento (plat. II). Resultados de pré-processamento adquiridos na plataforma II. O gráfico é similar ao Gráfico 1.

Conclusões Parciais

Os resultados mostraram que a fase inicial do algoritmo de construção requer uma fração do tempo total menor do que a fração correspondente à segunda fase do algoritmo. Aproximadamente, a primeira fase consumiu ¼ do tempo total, sobrando ¾ que foram consumidos pela segunda fase.

Em contraposição a essa relação constante que foi percebida, espera-se que, com o crescimento do número de células nas malhas de simulação, desde que a relação entre NC e NI seja mantida, o tempo consumido pela segunda fase tenda a prevalecer sobre o tempo consumido pela primeira fase, pois a complexidade da segunda fase é maior.

Considerando os objetivos a serem atingidos, a principal constatação dos testes realizados foi o ótimo desempenho obtido, alcançando menos de 7 segundos de processamento na plataforma I. Considerando malhas com complexidades semelhantes às testadas, esses resultados atendem satisfatoriamente à característica de visualização número 5, que exige uma pequena latência para o usuário antes de iniciar a visualização.

6.2.2 Consumo de Memória

O algoritmo de construção da estrutura de MR não possui boa localidade de memória. A cada passo do algoritmo, posições distantes do bloco de memória podem ser acessadas. Isto faz com que o *swap* seja intenso quando os dados não são totalmente acomodados na memória principal, prejudicando muito o desempenho do algoritmo. Dessa forma, é especialmente importante analisar a quantidade de memória consumida para especificar os requisitos mínimos necessários para um bom desempenho.

A Tabela 4 mostra as estruturas utilizadas pela implementação e seus respectivos consumos de memória. As estruturas *estr_célula* e *estr_nó_hier* são as mesmas introduzidas no Capítulo 5. A estrutura e*str_nó_hier*, entretanto, está modificada para prever possíveis informações que possam ser adicionadas à hierarquia de MR.

É válido salientar que o modelo original do reservatório foi representado por uma **tabela de vértices** e uma **tabela de células**. Por sua vez, o grafo de adjacência foi implementado através de uma **tabela de nós** e uma **tabela de arcos**. A tabela de nós associa a cada nó do grafo (*estr_nó_grafo*) referências para todos os arcos que incidem sobre ele e a tabela de arcos associa a cada arco (*estr_arco*) referências para seus nós extremos (Figura 6.2).

A análise do consumo de memória é feita em relação a NC, supondo-se que as células ativas estejam bem distribuídas espacialmente e que MAX_I, MAX_J e MAX_K possuam valores próximos. Com essas suposições, o número de arcos é aproximadamente igual a 3×NC e o grau de um nó do grafo de adjacência é igual a 6 no início do processamento, excetuando-se as células mais externas do reservatório.

Por ser um algoritmo de dizimação de células, o estado inicial do algoritmo é o que requer mais memória. Nesse momento, estão armazenados não só as tabelas de vértices e de células como também, em tamanho máximo, a tabela de arcos, a tabela de nós (com suas listas de 6 arcos incidentes) e o *heap*, além da floresta inicial e do

repositório pré-alocado de nós da hierarquia com seus (2×NC)-1 nós a serem preenchidos durante o laço de repetição. A utilização de um repositório de nós pré-alocados é feita para evitar que a alocação dinâmica desses nós prejudique o desempenho.

Tabela 4 - Estruturas utilizadas e seus respectivos consumos de memória.

Estrutura	Atributos	Bytes
estr_nó_grafo	- 6 índices para arcos[10] (*int*)	24
estr_arco	- erro geométrico (*float*) - índice no *heap* (*int*) - direção de colapso (*byte*) - 2 índices para os nós extremos (*int*)	17
estr_elem_heap	- erro geométrico (*float*) - índice cruzado para o arco (*int*) - grau do nó (*int*)	12
estr_vértice	- 3 coordenadas (*float*)	12
estr_célula	- 6 normais (*estr_vértice*) - 8 índices para vértices (*int*)	104
estr_AABB	- 2 *estr_vértice*	24
estr_nó_hier	- 8 informações adicionais (*int*) - direção de colapso (*byte*) - 3 ponteiros para filhos e pai (32 bits) - 1 erro geométrico (*float*) - 1 ponteiro para Célula (32 bits)	53
estr_nó_floresta	- 1 ponteiro para o nó raiz da hierarquia (32 bits)	4

Além disso, durante a construção da hierarquia, é necessário armazenar as macro-células associadas às raízes de cada árvore da floresta (as demais não são necessárias,

[10] Na verdade, a lista de arcos que compõe a estrela de um nó é dinâmica, variando à medida que os colapsos são realizados. O valor fixo, 6, está sendo utilizado para permitir o cálculo do consumo de memória no início do algoritmo.

pois podem ser reconstruídas posteriormente). Assim, no pior caso, estão armazenadas NC/2 macro-células durante a construção.

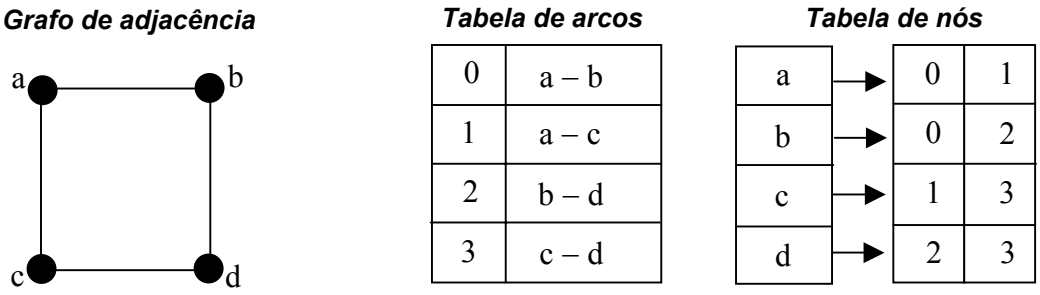


Figura 6.2- Esquema da implementação do grafo de adjacência.

A Tabela 5 mostra o consumo de memória no estado inicial do algoritmo e o consumo no pior caso gasto com as macro-células da floresta.

Tabela 5 - Consumo de memória durante a construção do algoritmo. Excetuando-se a última linha, todas representam o consumo no estado inicial.

	Bytes	
Tabela de vértices	12×(8×NC) = 96×NC	**Modelo original**
Tabela de células	104×NC	
Tabela de nós	24×NC	**Estado inicial da construção**
Tabela de arcos	17×(3×NC) = 51×NC	
Heap	12×(3×NC) = 36×NC	
Repositório de nós da hierarquia	53×2×NC	
Floresta inicial	4×NC	
Macro-células	104×(NC/2) = 52×NC	**Interior do laço de repetição**

Os cálculos foram realizados considerando a situação na qual não há compartilhamento de vértices entre células. Dessa forma, a tabela de vértices possui (8×NC) vértices, o que faz com que a quantidade de memória requerida para armazenar o modelo original (tabela de vértices + tabela de células) seja

$$(96\times NC) + (104\times NC) = (200\times NC)\ \textit{bytes}.$$

As estruturas de gerenciamento da construção (tabelas de arcos, nós e o *heap*) ocupam

$$(24\times NC) + (51\times NC) + (36\times NC) = (111\times NC)\ bytes.$$

E a hierarquia pré-alocada juntamente com a floresta inicial e suas macro-células ocupam

$$(53\times 2\times NC) + (4\times NC) + (52\times NC) = (162\times NC)\ bytes.$$

Portanto, a quantidade total de memória consumida pelo algoritmo de construção é

$$(200\times NC) + (111\times NC) + (162\times NC) = (473\times NC)\ bytes.$$

Isso significa que a quantidade de memória exigida para o algoritmo de construção é, aproximadamente, 136% maior do que a quantidade de memória necessária para armazenar apenas o modelo original.

Para finalizar a análise, é necessário verificar o quanto de memória é consumido durante a visualização. Após o algoritmo de construção, a hierarquia de células hexaédricas está vazia, sem células (apenas as relativas às folhas) e sem *aabbs*. É preciso, então, percorrer a árvore de baixo para cima, recalculando as macro-células e calculando as *aabbs* de cada nó. Para isto, a informação que indica a direção de colapso é fundamental, pois basta aplicar o padrão de colapso correspondente para obter a geometria correta da macro-célula. Todo esse processo é feito em tempo linear e está embutido nas tomadas de tempo realizadas na seção anterior.

Após essa reconstrução, a hierarquia de células é composta por $(2\times NC)-1$ nós da estrutura, NC-1 macro-células e NC-1 *aabbs* e, portanto, o consumo do modelo original somado ao da hierarquia é de aproximadamente

$$(200\times NC) + (53\times 2\times NC) + (104\times NC) + (24\times NC) = (434\times NC)\ bytes.$$

Ou seja, a hierarquia de células hexaédricas requer aproximadamente 120% a mais de memória em relação ao modelo original.

A Tabela 6 mostra os valores de consumo de memória que seriam gastos em cada malha segundo as estimativas teóricas e os valores reais de consumo medidos durante a execução do programa. Estes foram adquiridos de forma aproximada através da observação do gerenciador de memória do Windows 2000. Os consumos reais relativos ao modelo original não correspondem exatamente aos valores esperados pela análise teórica, pois, dentre outros fatores, o leitor dos dados de simulação armazena, adicionalmente, alguns cabeçalhos com informações descritivas a respeito das propriedades e das grandezas físicas da simulação, sendo essas informações consideradas no consumo do modelo original.

Tabela 6 – Valores aproximados, em MB, relativos ao consumo de memória estimado e ao consumo de memória realmente gasto durante a execução do programa medidos através do gerenciador de memória do Windows 2000.

	Cálculo estimado de consumo (em MB)			Consumo real (em MB aproximadamente)		
	Modelo original (M)	Construção da hierarquia, desconsiderando M	Hierarquia de células hexaédricas	Modelo original (M')	Construção da hierarquia, desconsiderando M'	Hierarquia de células hexaédricas
A_bc	0,706	0,964	0,826	1,0	0,9	1,0
B_cp	1,722	2,351	2,015	2,5	2,9	2,5
I_bc	2,26	3,085	2,644	3,2	4,8	3,0
D_cp	6,01	8,203	7,031	7,1	11,0	7,9
E_bc	8,023	10,95	9,38	13,2	17,0	10,5
F_bp	9,7	13,26	11,37	14,0	19,2	12,0

Conclusões Parciais

O estudo efetuado permite inferir a quantidade mínima de memória exigida pelo MMR para garantir um bom desempenho. No estudo teórico, a memória adicional exigida é aproximadamente 136% maior do que a quantidade necessária para armazenar o modelo original. Nos testes práticos, percebem-se algumas variações, o pior caso sendo de 154% de memória adicional (Tabela 7- coluna 4).

Outro valor importante é o do consumo de memória da hierarquia de células hexaédricas. O estudo teórico indicou um índice de aproximadamente 120% de memória adicional, mas, na prática, o pior caso foi de 111% (Tabela 7- coluna 6).

Tabela 7 – Valores aproximados, em MB, relativos ao consumo real de memória, juntamente com a medida comparativa do custo adicional.

	Modelo original (M')	Construção da hierarquia desconsiderando M' (C)	Relação entre C e M' (%)	Hierarquia de células hexaédricas (H)	Relação entre H e M' (%)
A_bc	1,0	0,9	**90%**	1,0	**100%**
B_cp	2,5	2,9	**116%**	2,5	**100%**
I_bc	3,2	4,8	**150%**	3,0	**93%**
D_cp	7,1	11,0	**154%**	7,9	**111%**
E_bc	13,2	17,0	**129%**	10,5	**79%**
F_bp	14,0	19,2	**137%**	12,0	**85%**

6.3 Extração de Malhas

Esta seção tem por objetivo analisar a qualidade das aproximações geradas pelo MMR e a eficiência dos algoritmos de extração. Inicialmente, a qualidade das aproximações é medida na seção Erro Geométrico nas Aproximações. Posteriormente, os índices de redução da complexidade das malhas são analisados na seção Erro Projetado e Redução de Complexidade e, finalmente, a eficiência dos algoritmos é medida na seção Taxas de Interatividade.

6.3.1 Erro Geométrico nas Aproximações

A qualidade das aproximações geradas é medida, nesta seção, através do erro geométrico armazenado pela estrutura de MR. Como foi visto, este erro representa uma aproximação da distância entre o modelo aproximado e o original.

O Gráfico 3 analisa a variação do número de células nas aproximações em função dos erros geométricos, que foram normalizados entre 0% e 20% da diagonal da caixa envolvente do modelo. As aproximações foram geradas pelo algoritmo de extração dependente do erro geométrico.

Excetuando-se a malha I_bc, todas as malhas conseguem ser reduzidas em até 34% sem que o erro geométrico introduzido ultrapasse o valor de 0,0002.

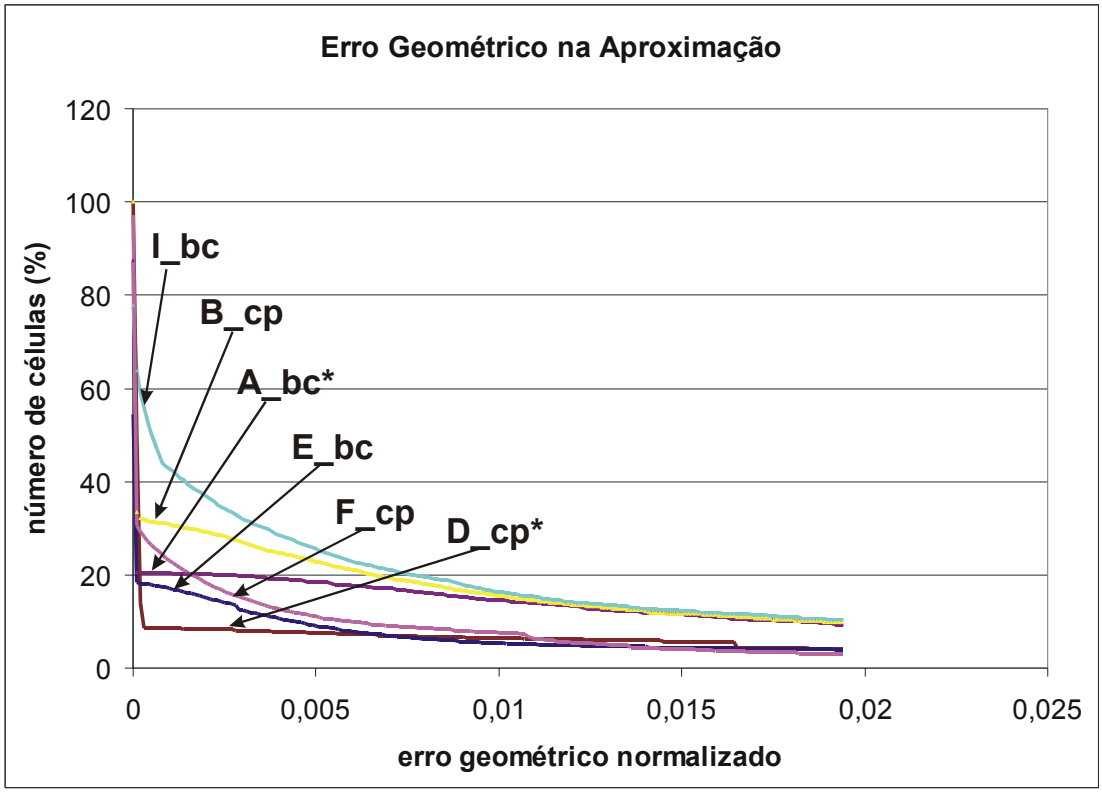
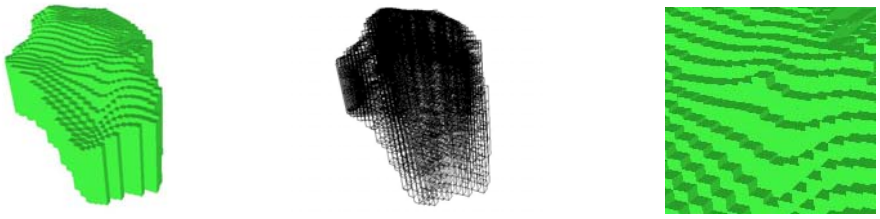


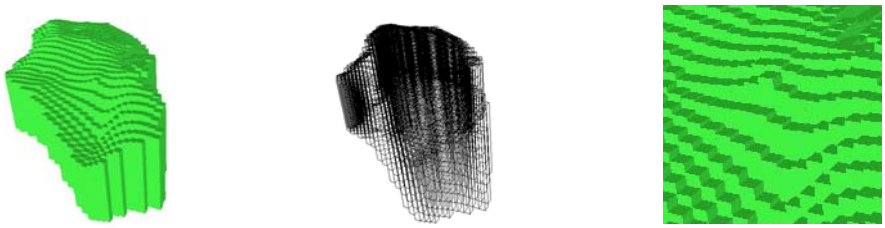
Gráfico 3 – Erro geométrico nas aproximações. Resultados obtidos pelo algoritmo de extração dependente do erro geométrico submetido a consecutivas parametrizações de erro. Os erros geométricos estão normalizados entre 0% e 20% das diagonais das caixas envolventes dos reservatórios.

Para melhor visualizar o erro normalizado utilizado, a Figura 6.3 e a Figura 6.4 ilustram o processo de degradação que ocorre com as malhas A_bc e F_cp. Na A_bc, percebe-se que a malha pode ser representada sem erro geométrico com 87% de suas faces originais e com erro muito pequeno com 20,5%. A partir desta etapa, a redução da malha acarreta a inserção de erros mais significativos. Um fenômeno análogo ocorre com o exemplo F_cp, porém o erro cresce numa taxa menor após a fase inicial

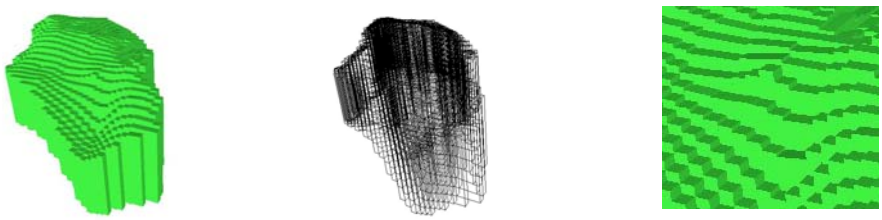
de super-amostragem. No Gráfico 3, pode-se perceber esta diferença entre as taxas de crescimento do erro.



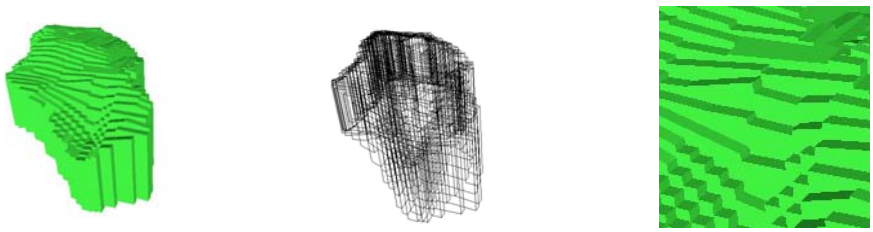
(a) 0,0 de erro geométrico e 87% do total de faces



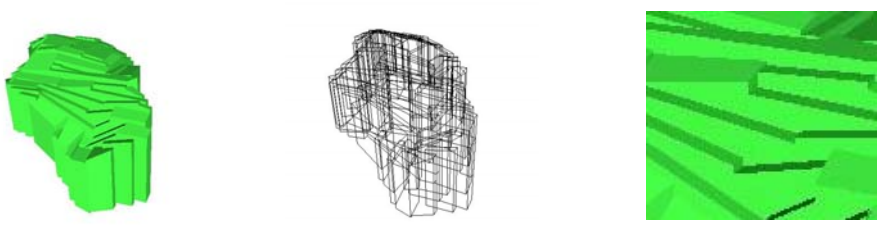
(b) 0,0001 do erro geométrico e 20,5% do total de faces



(c) 0,01 do erro geométrico e 14,4% do total de faces



(d) 0,03 do erro geométrico e 6,4% do total de faces



(e) 0,07 do erro geométrico e 1,9% do total de faces

Figura 6.3 - Degradação da malha A_bc. Na coluna central, o respectivo desenho em *wireframe* e, na coluna à direita, a ampliação de uma pequena região do reservatório para possibilitar a melhor percepção do erro introduzido.

(a) 0,0 de erro geométrico e 97,3% do total de faces

(b) 0,0001 do erro geométrico e 30,5% do total de faces

(c) 0,01 do erro geométrico e 7,6% do total de faces

(d) 0,03 do erro geométrico e 1,9% do total de faces

(e) 0,07 do erro geométrico e 0,7% do total de faces

Figura 6.4 - Degradação da malha F_cp. Na coluna central, o respectivo desenho em *wireframe* e, na coluna à direita, a ampliação de uma pequena região do reservatório para possibilitar a melhor percepção do erro introduzido.

A Figura 6.5 ilustra em detalhe as aberturas e as interseções introduzidas nas aproximações. Elas são bastante evidentes nas aproximações de baixa resolução.

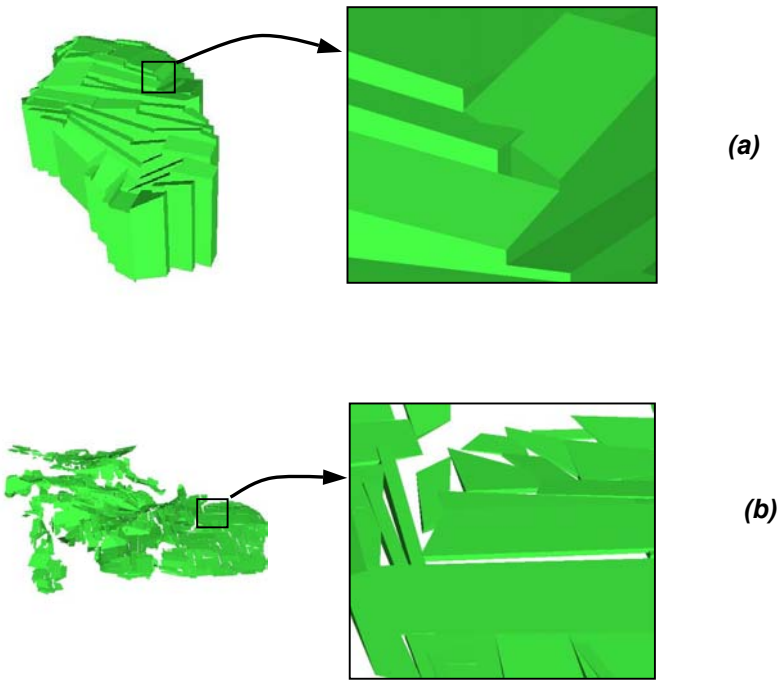


Figura 6.5 – Aberturas e interseções criadas nas aproximações geradas pelo MMR proposto neste trabalho.

Conclusões Parciais

Primeiramente, o fato das malhas poderem ser reduzidas drasticamente sem a inserção de erros expressivos indica que elas estão super-amostradas, o que não é uma surpresa, visto que a geometria das malhas é presa às malhas de referência, não possuindo curvatura nas colunas de células dispostas em profundidade. Nessas colunas, o único fator que pode impedir uma redução sem erro das taxas de amostragem é o possível espaçamento existente entre camadas consecutivas.

As reduções obtidas com as malhas D_cp e E_bc, que possuem os maiores números de camadas (MAX_K é igual a 23 e 126, respectivamente), são fortes evidências de que a restrição à malha de referência realmente influencia a capacidade de redução sem erro, uma vez que as reduções nessas malhas obtiveram os menores valores. Portanto, há uma grande probabilidade de que as taxas de redução obtidas variem proporcionalmente ao número de camadas das malhas.

Além disso, as ilustrações mostraram que a inserção de aberturas e interseções nas aproximações faz com que, gradativamente, as malhas percam suas características geométricas iniciais, de forma que as aproximações de baixa resolução são formadas por um conjunto desestruturado de células.

6.3.2 Erro Projetado e Redução de Complexidade

O objetivo desta seção é analisar a redução da complexidade das aproximações geradas pelo algoritmo de extração dependente da câmera, utilizando-se, em particular, uma câmera em perspectiva. Uma vez fixado um limite máximo de erro projetado permitido (parâmetro do algoritmo), em *pixels*, esta redução é determinada pela distância à câmera (quanto mais distante, maior é a redução) e pelo descarte das regiões da malha que estão fora do volume de visão.

A Figura 6.6 relaciona, perceptualmente, diferentes parametrizações de erro projetado à similaridade de aparência. Essa relação é válida somente para a resolução de tela (*viewport*) que foi utilizada, 1000x800 *pixels*. Na coluna da esquerda, há imagens geradas pelo algoritmo a partir de um único ponto de vista sob diferentes parametrizações. Na coluna central, estão suas imagens **diferença**[11] em relação à imagem gerada pela resolução mais alta do reservatório sob o mesmo ponto de vista e mesma resolução de tela. Na coluna da direita, está assinalado o parâmetro de erro utilizado. Com o aumento do erro projetado, há também o aumento do erro de similaridade[12], sendo que é importante notar que o erro é visualmente pequeno até a parametrização de 1.0 *pixel.*

---

[11] *Uma imagem* diferença *é aquela cujos pixels são dados pelo módulo da diferença, pixel a pixel, entre duas imagens quaisquer de mesmo tamanho.*

[12] *O erro de similaridade entre duas imagens I_1 e I_2 de mesmas dimensões é dado pela média dos pixels da imagem* diferença *entre elas.*

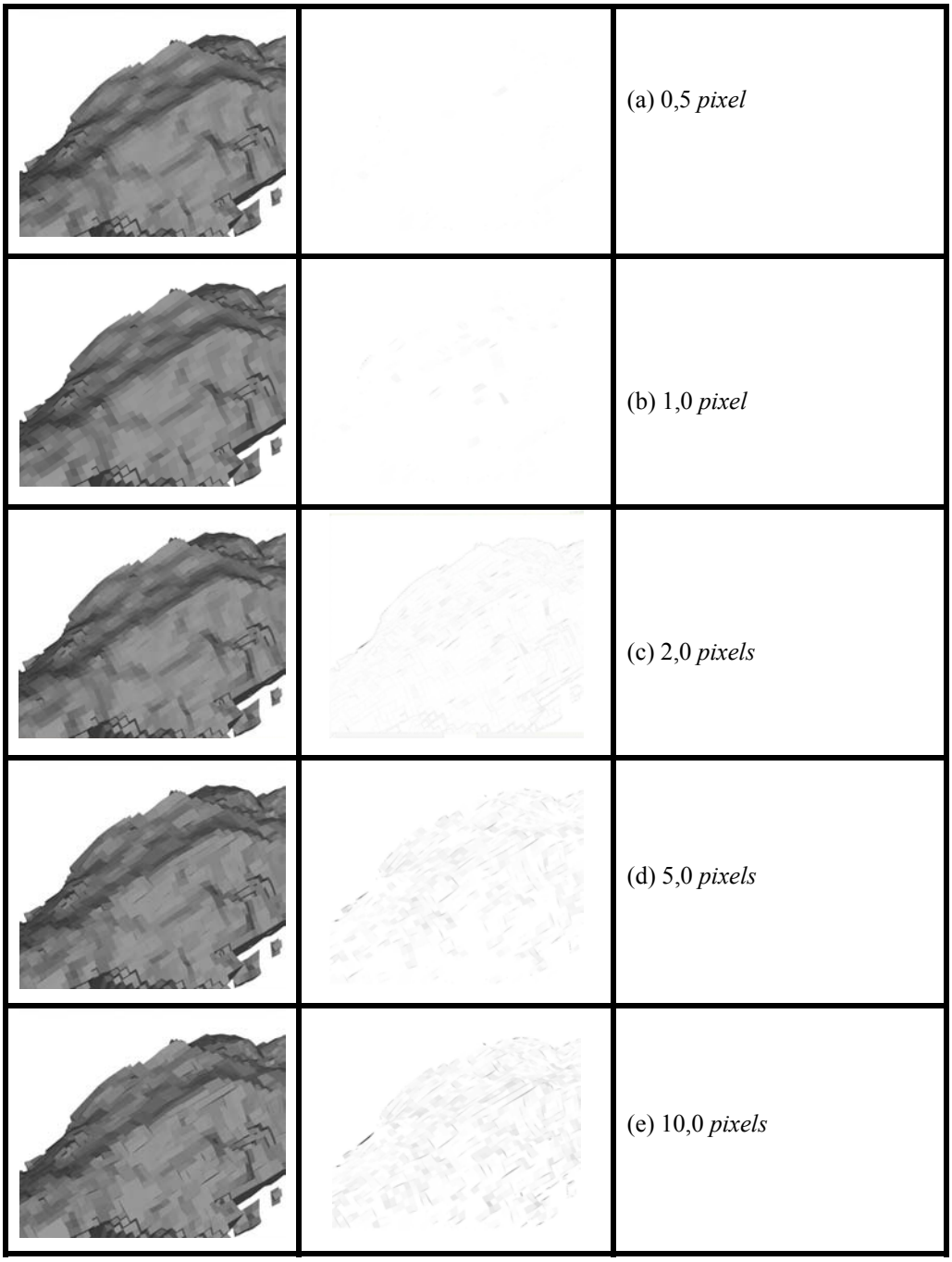


Figura 6.6 – Similaridade de aparência. Na coluna à esquerda, as imagens geradas a partir das aproximações. Na coluna central, suas imagens diferença em relação à imagem gerada pela maior resolução do reservatório. Na coluna à direita, os erros utilizados como parâmetros para a geração das aproximações. As imagens foram geradas com a resolução 1000x800 *pixels*.

Bateria de testes

Para realizar os testes, foram traçados quatro tipos de caminho para cada malha. A Tabela 8 descreve alguns dados sobre eles. O campo "marcas em destaque" se refere a

pontos específicos do caminho que são vistos no decorrer deste texto e o campo "quantidade de quadros" indica quantos quadros são gerados durante o percurso. Os caminhos classificam-se em: **Básico 1** (**B1**), **Básico 2** (**B2**), **Básico 3** (**B3**) e **Livre 1** (**L1**).

A Figura 6.7, a Figura 6.8 e a Figura 6.9 ilustram os caminhos básicos para o caso de teste F_cp. Os caminhos B1, aproximadamente sob a vista superior (perpendicular e acima do plano formado pelos eixos *x* e *y*), realizam apenas uma aproximação seguida de um afastamento com o foco de visão sempre voltado para o centro do reservatório.

Os caminhos B2 utilizam uma vista lateral e são semelhantes aos B1, fazendo também uma aproximação seguida de um afastamento. Por sua vez, os caminhos B3, sob uma vista superior inclinada, realizam aproximações, translações perpendiculares à direção de visão e afastamentos curtos. Vale observar que estes caminhos foram traçados de forma semelhante em todos casos de teste, ao contrário dos caminhos L1, que efetuam passeios livres pelos reservatórios.

Para efetuar os testes, cada caminho foi percorrido várias vezes com diferentes parametrizações do erro projetado, sempre utilizando a resolução de tela de 1000x800 *pixels*. Em vez de manter o tempo de cada percurso constante, interpolando as posições da câmera ao longo do caminho de forma que a velocidade seja sempre a mesma, optou-se por fixar o número de posições da câmera e deixar que o tempo gasto no percurso varie. Isto justifica o campo "número de quadros" da Tabela 8.

Tabela 8 – Caminhos. Cada linha da tabela representa um caminho distinto. Existem quatro caminhos para cada malha de teste classificados de acordo com seu tipo.

	Tipo	Quantidade de quadros	Marcas em destaque
A_bc	B1	848	A e B
	B2	3080	A e B
	B3	1253	
	L1	4787	
B_cp	B1	2590	A e B
	B2	2810	A e B
	B3	1757	
	L1	3905	
I_bc	B1	3112	A e B
	B2	2772	A e B
	B3	1388	
	L1	3177	
D_cp	B1	808	A e B
	B2	2481	A e B
	B3	1552	
	L1	3469	
E_bc	B1	664	A e B
	B2	1892	A e B
	B3	500	
	L1	1429	
F_cp	B1	1369	A e B
	B2	826	A e B
	B3	779	
	L1	2329	

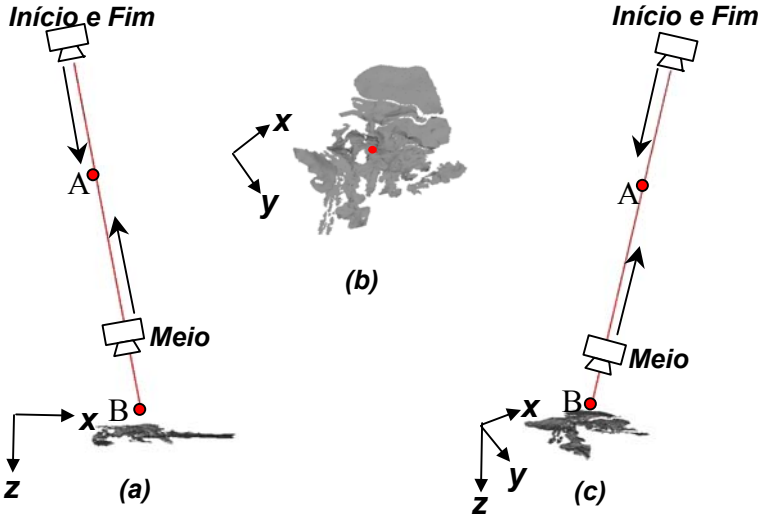


Figura 6.7 – Caminho B1 da malha F_cp. Em (a) e (c), vistas perpendiculares à direção principal da câmera. Em (b), a vista superior do reservatório.

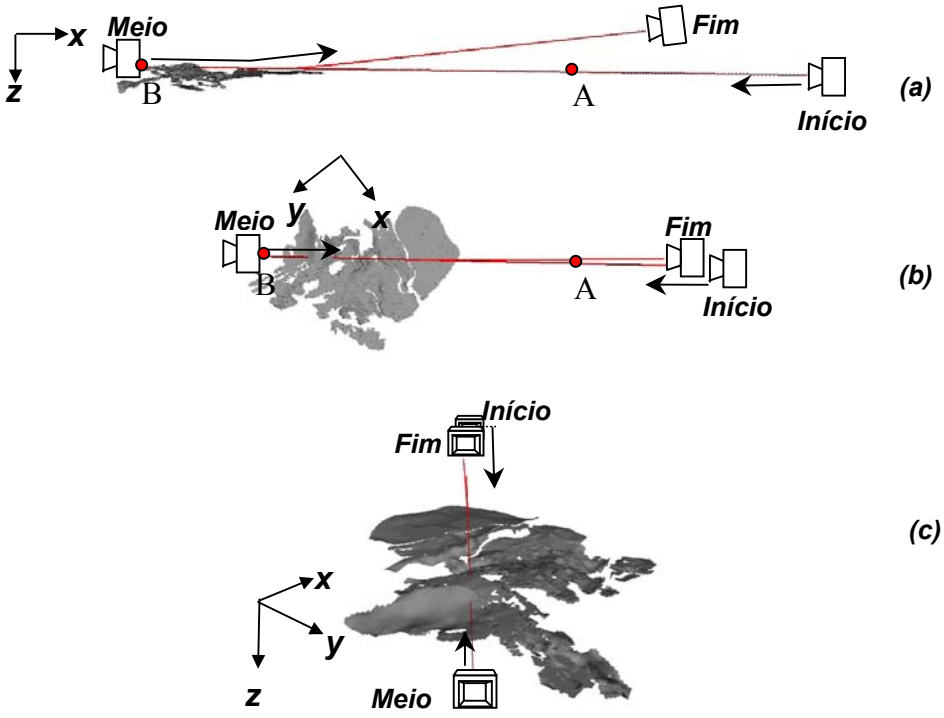


Figura 6.8 – Caminho B2. Em (a), a vista lateral à direção principal da câmera. Em (b), a vista superior do reservatório e, em (c), a uma vista quase frontal à direção de deslocamento da câmera.

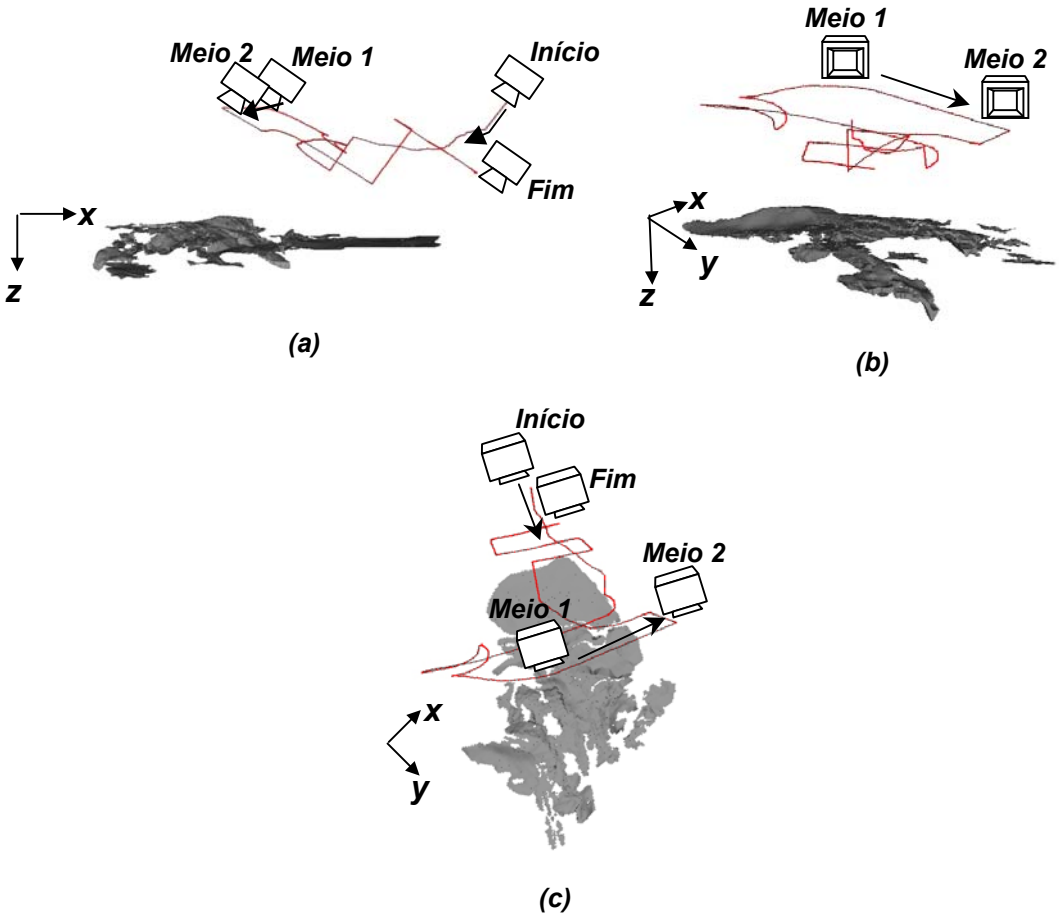


Figura 6.9 – Caminho B3. Em (a), uma vista perpendicular à câmera, destacando o início, o meio e o fim do percurso. Em (b), uma vista quase frontal à câmera, destacando-a em dois pontos intermediários do percurso. Em (c), a vista superior do reservatório, destacando a câmera no início, no meio e no final do percurso.

Com o intuito de tornar a análise mais clara, os caminhos serão analisados separadamente.

Caminhos B1 e B2

Estes caminhos foram traçados especificamente para analisar (a) a variação da redução de complexidade à medida que o reservatório ocupa uma área gradualmente

maior na tela e (b) a influência conjunta do descarte e da projeção nessa variação. Por isso, justifica-se o traçado controlado e simples dos dois percursos.

Foram destacados dois pontos de referência: A e B. Até o ponto A, a redução da complexidade é determinada somente pela projeção do erro. À medida que a câmera se aproxima do reservatório, sua área na imagem vai aumentando até que, exatamente em A, o reservatório ocupa o maior espaço na imagem sem que ocorra nenhum descarte (Figura 6.10). A partir daí, entre os pontos A e B, o comportamento é determinado tanto pela projeção do erro como pelo descarte, o primeiro requisitando o aumento das células e segundo eliminando-as. Do ponto B em diante, a câmera faz o percurso contrário, afastando-se.

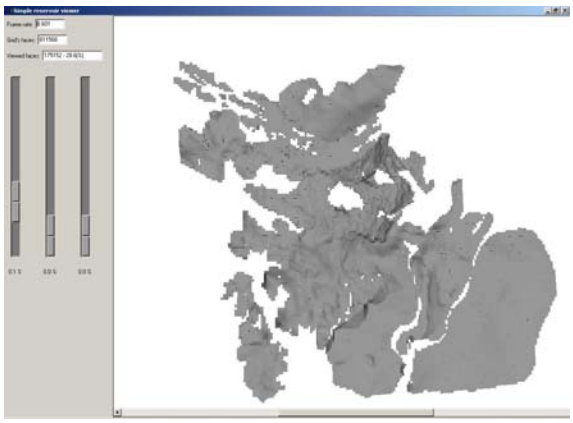
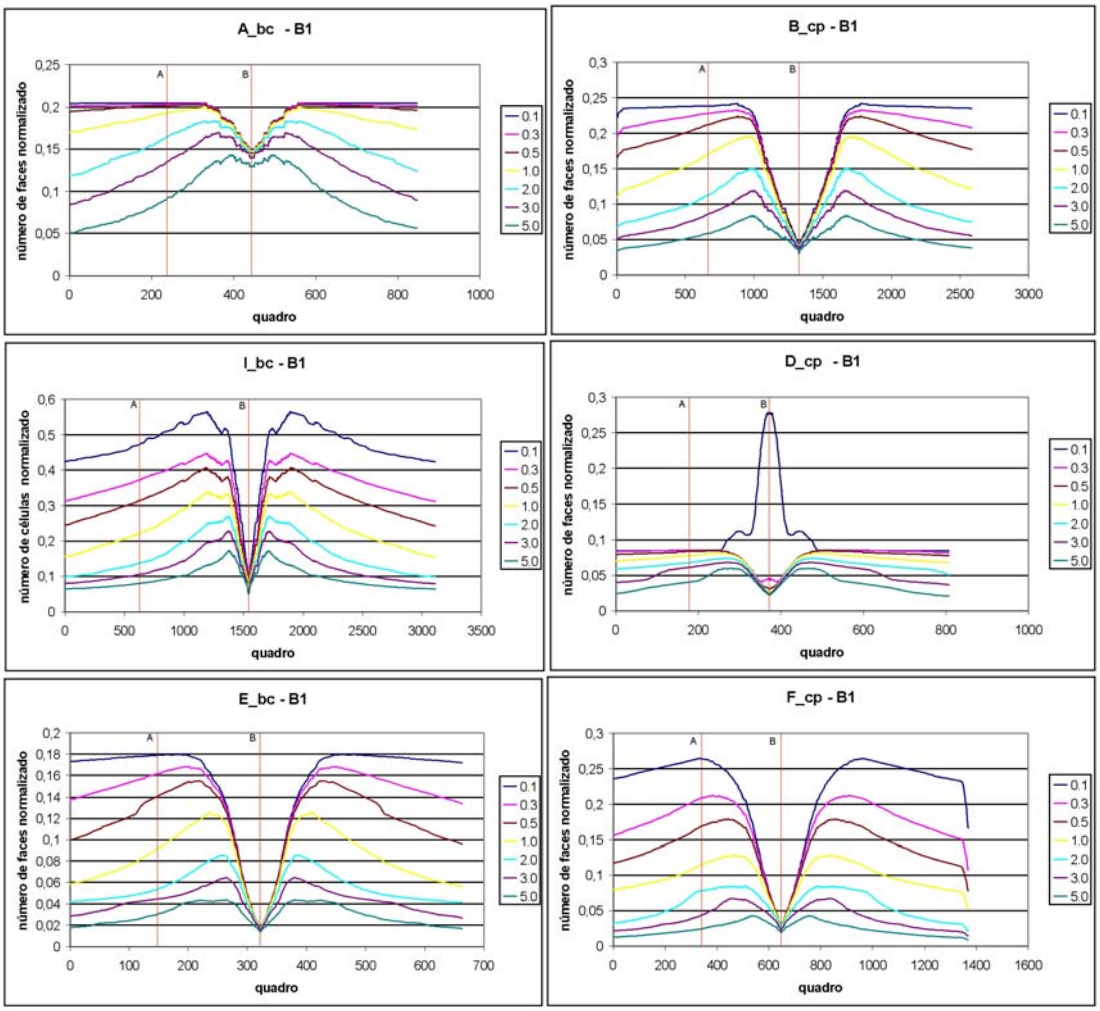


Figura 6.10 – Ampliação máxima. Imagem gerada no ponto A no passeio realizado sobre F_cp com erro de 0,1 *pixel*. O reservatório está quase totalmente ajustado ao tamanho da tela.

O Conjunto de Gráficos 1 refere-se aos caminhos B1 e mostram a quantidade de células que estão sendo desenhadas a cada quadro. Cada curva representa uma parametrização do erro, em *pixels*, considerado no percurso. Os piores resultados ocorrem na malha I_bc, cuja complexidade atingiu aproximadamente 55% para uma parametrização de 0,1 *pixel*. Nos demais casos, percebem-se resultados mais próximos, sendo que, coerentemente com a seção anterior, quanto maior o número de células, maiores são as taxas de redução alcançadas.

Na região do gráfico entre os pontos A e B, percebe-se a compensação entre o refinamento e o descarte. Na parte crescente da curva, o refinamento é dominante,

mas, à medida que a região que está dentro do volume de visão vai se aproximando de sua resolução máxima, o descarte vai se tornando dominante até que a curva passa a ser decrescente. Há apenas uma exceção[13] a este comportamento: a curva 0.1 do gráfico D_cp do Conjunto de Gráficos 1. Nela, o refinamento é dominante em quase toda a região entre A e B.



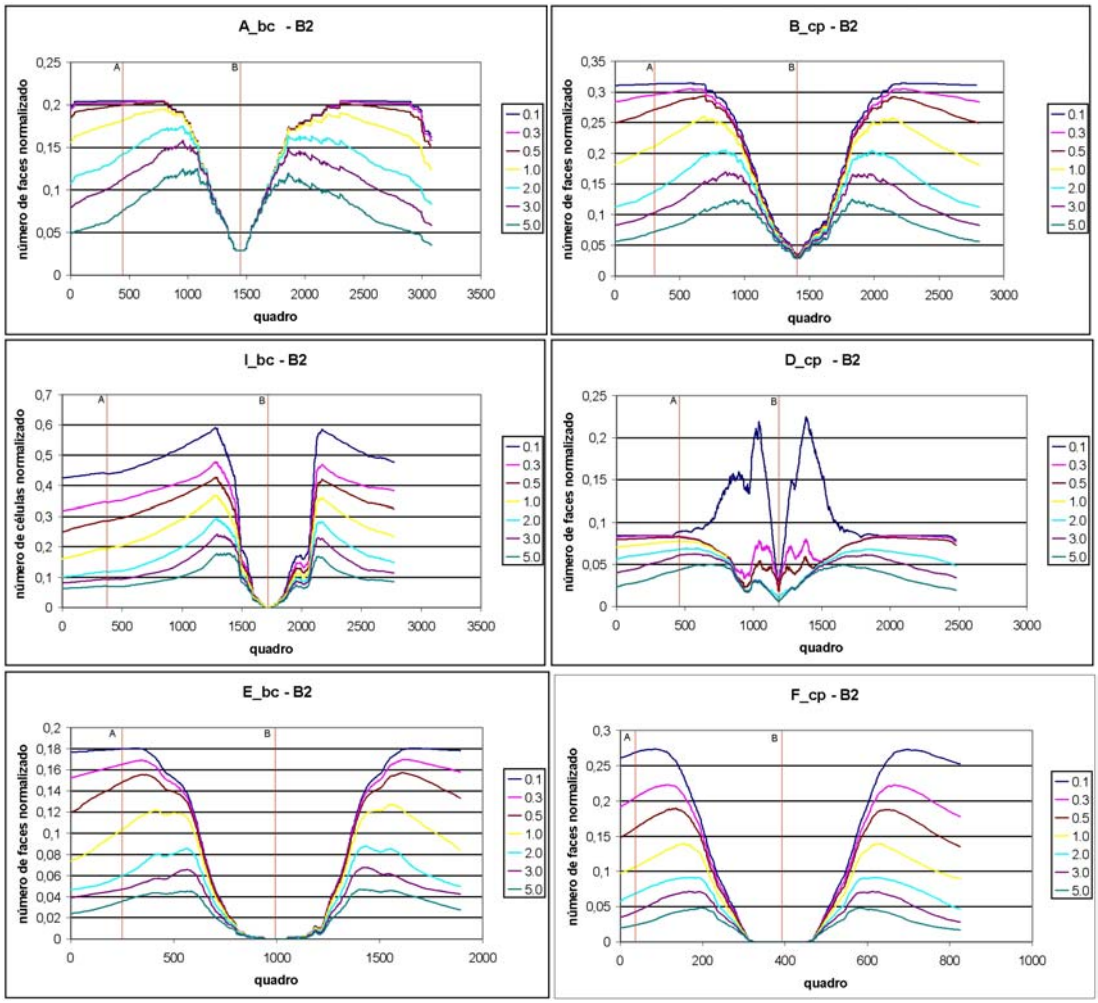
Conjunto de Gráficos 1 – Variação do número de células nas aproximações ao longo dos quadros (B1). Cada gráfico representa o caminho B1 em um reservatório diferente.

Uma vez que os caminhos B2 são semelhantes aos B1, mudando-se apenas a direção de deslocamento da câmera em relação à malha, seus resultados, que estão ilustrados

---

[13] A exceção existente na malha D_cp pode ser explicada através do Gráfico 3, que representa o crescimento do erro geométrico nas aproximações. Nele, percebe-se a formação de uma espécie de patamar na malha D_cp. O que ocorre na curva 0.1 dos caminhos B1 e B2 é a necessidade de ultrapassar esse patamar em algum momento durante a aproximação, aumentando abruptamente o número de células necessário para atender ao limite máximo de erro permitido.

no Conjunto de Gráficos 2, são a contra-prova dos resultados obtidos em B1. Analisando-os, percebe-se que realmente os comportamentos são parecidos, oferecendo mais confiabilidade aos resultados obtidos.



Conjunto de Gráficos 2 - Variação do número de células nas aproximações ao longo dos quadros (B2). Cada gráfico representa o caminho B2 em um reservatório diferente.

Caminhos B3

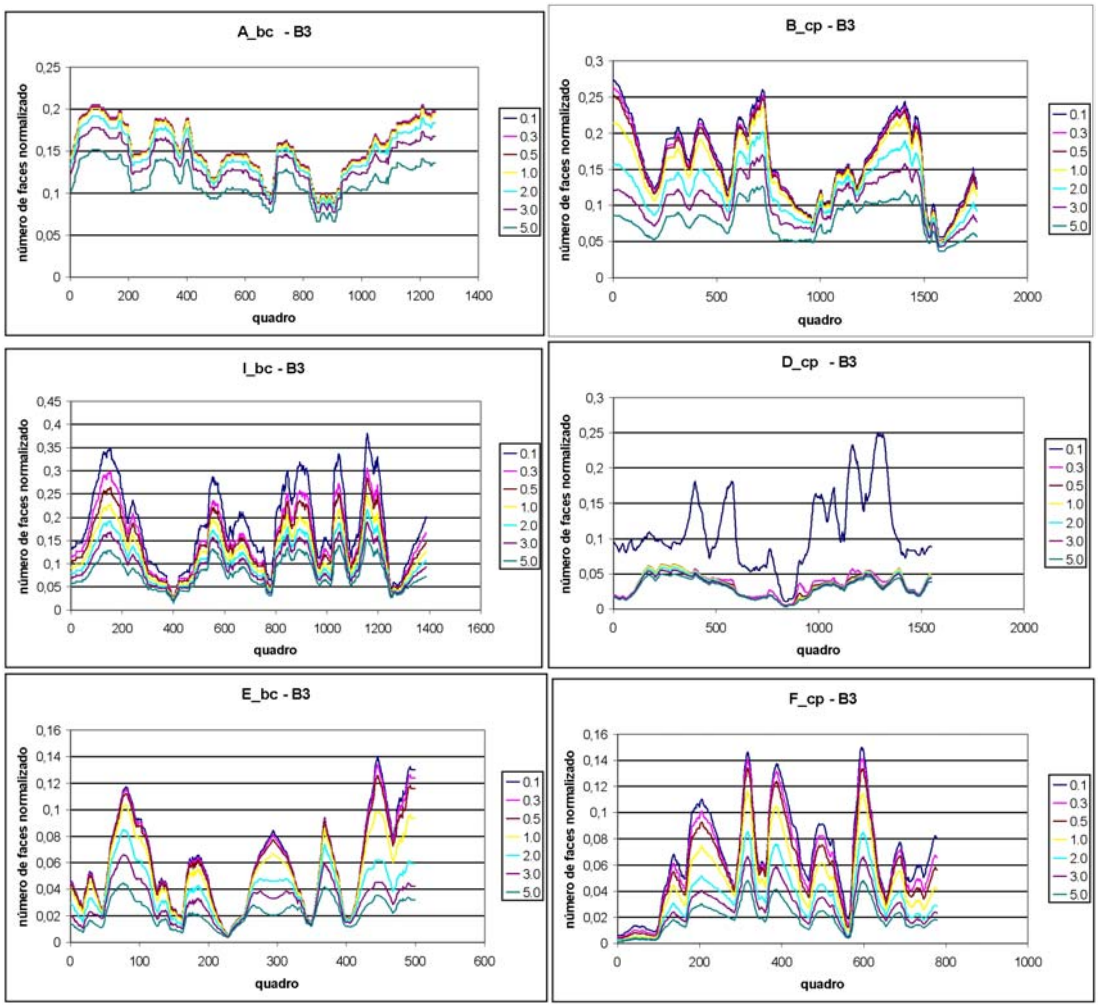
Os caminhos B3 foram projetados para possibilitar a avaliação do descarte como fator determinante para a redução da complexidade. Procurou-se traçá-los de forma que, em grande parte do percurso, apenas uma pequena região do reservatório estivesse visível.

O Conjunto de Gráficos 3 representa os resultados obtidos para os caminhos B3. Os pontos de mínimo das curvas correspondem a pontos nos quais apenas uma pequena

parte do reservatório está dentro do volume de visão. Esses pontos são nítidos em todos os casos de testes, inclusive no I_bc. Percebe-se que as curvas se aproximam nesses pontos, indicando que o descarte, como é de sua natureza, independe da parametrização de erro considerada.

Apesar da influência do descarte ser mais notável na malha I_bc (no sentido de que a complexidade da malha cai de aproximadamente 40% para menos de 5%), é neste caso de teste que obtêm-se os piores resultados de redução, o que ilustra o fato conhecido de que a técnica de descarte sozinha não garante baixas taxas médias de redução.

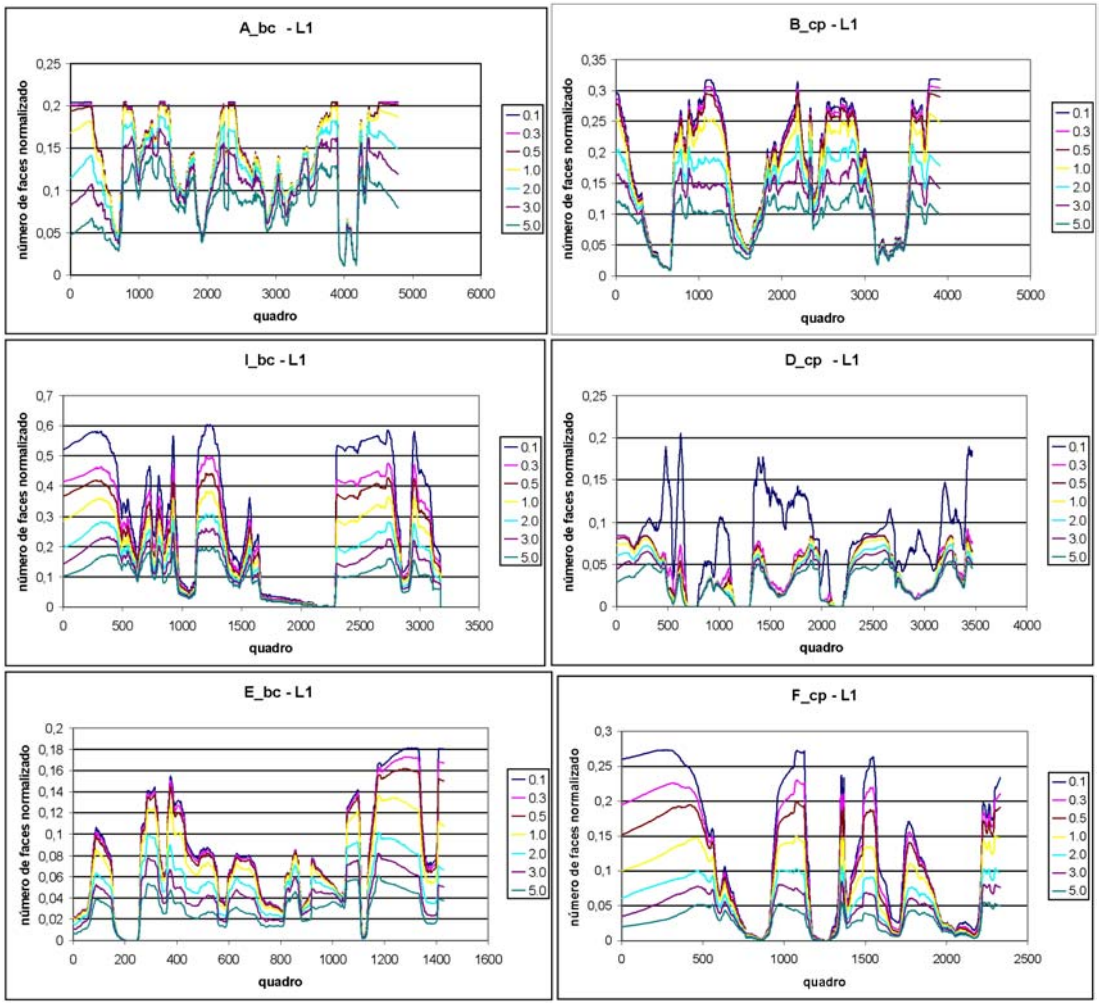
Mais uma vez percebe-se que as maiores malhas apresentam resultados melhores.



Conjunto de Gráficos 3 - Variação do número de células nas aproximações ao longo dos quadros (B3). Cada gráfico representa o caminho B3 em um reservatório diferente.

Caminhos L1

Os caminhos L1 exploram ângulos de visão variados sob os quais os reservatórios podem ser observados durante uma inspeção real, objetivando detectar os piores casos de redução de complexidade. Mais uma vez, o pior caso é o I_bc, que chega a ser representado por 60% de sua complexidade quando a parametrização é de 0,1 *pixel*. Nos demais, a complexidade máxima alcançada é limitada em 32% para a mesma parametrização de erro.



Conjunto de Gráficos 4 – Variação do número de células nas aproximações ao longo dos quadros (L1). Cada gráfico representa o caminho L1 em um reservatório diferente.

Conclusões Parciais

Considerando a resolução de tela utilizada nos testes (1000x800), pode-se assumir que as parametrizações de 0,1 *pixel* e 0,5 *pixel* estão associadas a um baixo erro de

similaridade de aparência (ver Figura 6.6). Assim, as correspondentes curvas 0.1 e 0.5 nos gráficos caracterizam os valores de complexidade alcançados pelo MMR sem que praticamente nenhum erro visual seja cometido. As curvas 0.5 possuem seu pior caso na malha I_bc: em torno de 45% de complexidade (caminhos B2 e L1). Nas demais malhas, o pior caso foi em torno de 30%. E, especificamente, nas três malhas mais densas, o pior caso foi de 20%.

Nos caminhos B1 e B2, o valor de complexidade alcançado no ponto A é importante, pois este ponto corresponde à posição do percurso que permite a visualização da malha por inteiro e com ampliação (*zoom*) máxima, o que, na visualização, é uma situação típica de manipulação. Para as curvas 0.5, o pior caso foi de aproximadamente 40% na malha I_bc. Nas demais, o valor alcançado foi em torno de 27%, sendo que, particularmente nas três malhas mais densas, o valor foi de 17%.

Os valores obtidos com a malha I_bc são importantes para a averiguação do comportamento do MMR proposto sob condições desfavoráveis; entretanto, eles não representam bem a potencialidade do MMR na prática usual. Assim, os mais representativos são os valores encontrados nas demais malhas. A Tabela 9 e a Tabela 10 reportam os valores representativos obtidos, separando as malhas quanto à densidade de células:

- baixa densidade: A_bc e B_cp;
- alta densidade: D_cp, E_bc e F_cp.

Um fato importante a ser observado é que, na Seção 6.3.1, são obtidas malhas de boa qualidade geométrica com complexidade menor do que 30% do número original de células, sendo esperado que, a partir dessas malhas, fossem geradas imagens com boa similaridade de aparência. Essa expectativa foi confirmada através dos resultados aqui obtidos e, de certa forma, isso valida o erro geométrico utilizado pelo MMR.

Finalizando, a análise aqui realizada foi bastante conservadora e não explorou os resultados obtidos nas curvas que representam erros projetados maiores do que 1,0 *pixel*. Entretanto, as malhas geradas com esses parâmetros de erro podem ser úteis em algum caso específico de visualização que não exija alta similaridade de aparência.

Tabela 9 - Relatório com as complexidades alcançadas nos piores casos, levando em consideração todos os caminhos percorridos.

	Complexidade aproximada nos piores casos	Parametrização de erro utilizada
Baixa densidade	30%	0,5
Alta densidade	20%	0,5
Baixa densidade	25%	1,0
Alta densidade	15%	1,0

Tabela 10 - Relatório com as complexidades alcançadas nos piores casos, levando em consideração apenas os valores no ponto A dos caminhos B1 e B2.

	Complexidade aproximada nos piores casos	Parametrização de erro utilizada
Baixa densidade	27%	0,5
Alta densidade	17%	0,5
Baixa densidade	17%	1,0
Alta densidade	12%	1,0

6.3.3 Taxas de Interatividade

Para completar a avaliação feita na seção anterior, nesta seção será analisado o desempenho do algoritmo de extração dependente da câmera. Através dessa medição, pode-se avaliar realmente o ganho obtido com a utilização do MMR, verificando se a redução de complexidade das aproximações é suficiente e se os cálculos realizados para efetuar essa redução – cálculo de descarte e de projeção de erro, além do caminhamento pela hierarquia – são compensatórios.

A análise é feita através da medida **quadros por segundo** (**qps**), que foi calculada instantaneamente através da fórmula:

$$Qps(q)=\frac{1}{\frac{1}{10}\sum_{i=0}^{9} T(q-i)};$$

onde q representa um quadro e $T(q)$ é o tempo gasto, em segundos, para processá-lo.

A medida qps é altamente dependente da plataforma de *hardware* utilizada, principalmente de seu processador gráfico. Assim, foram utilizadas duas plataformas diferentes nos testes, a III e a IV, e a implementação fez uso dos recursos de seus processadores gráficos. Dessa forma, parte do processamento está sendo feito pela CPU, enquanto outra parte está sendo feita pela **GPU** (Unidade de Processamento Gráfico). Por isso, é necessário marcar o tempo de "relógio de parede" gasto para processar um quadro e tomar o cuidado de evitar que processos externos concorram com a aplicação de visualização.

Para realizar os testes, foram escolhidas as malhas B_cp e F_cp, uma de baixa e outra de alta densidade de células, respectivamente. Os testes consistem em percorrer o caminho L1, na resolução de tela 1000x800, com quatro parâmetros diferentes de erro projetado (0,1, 0,5, 1,0 e 2,0 *pixels*), anotando a taxa de qps obtida. Para facilitar a análise, o Gráfico 4 e o Gráfico 6 mostram o número de células desenhadas a cada quadro durante o percurso realizado sobre as malhas B_cp e F_cp, respectivamente.

O Gráfico 5 e o Gráfico 7 representam os resultados obtidos na plataforma IV com valores truncados em 140 qps. Na malha B_cp, a curva "sem MMR", que representa o caminho percorrido sem a utilização de técnicas de aceleração, obteve uma média de 12,97 qps, mantendo-se praticamente constante no decorrer do percurso. Por outro lado, a utilização do MMR possibilitou uma taxa de interatividade com o valor mínimo de 33,85 qps na curva 0.1.

Na malha F_cp, a curva "sem MMR" obteve uma média de 2,4 qps, enquanto os piores casos das curvas 0.1 e 0.5, que utilizam o MMR, foram de 7,58 qps e 11,59 qps, respectivamente.

Como esperado, as curvas que representam as taxas de interatividade possuem o comportamento inverso às respectivas curvas que representam o número de células

desenhadas.

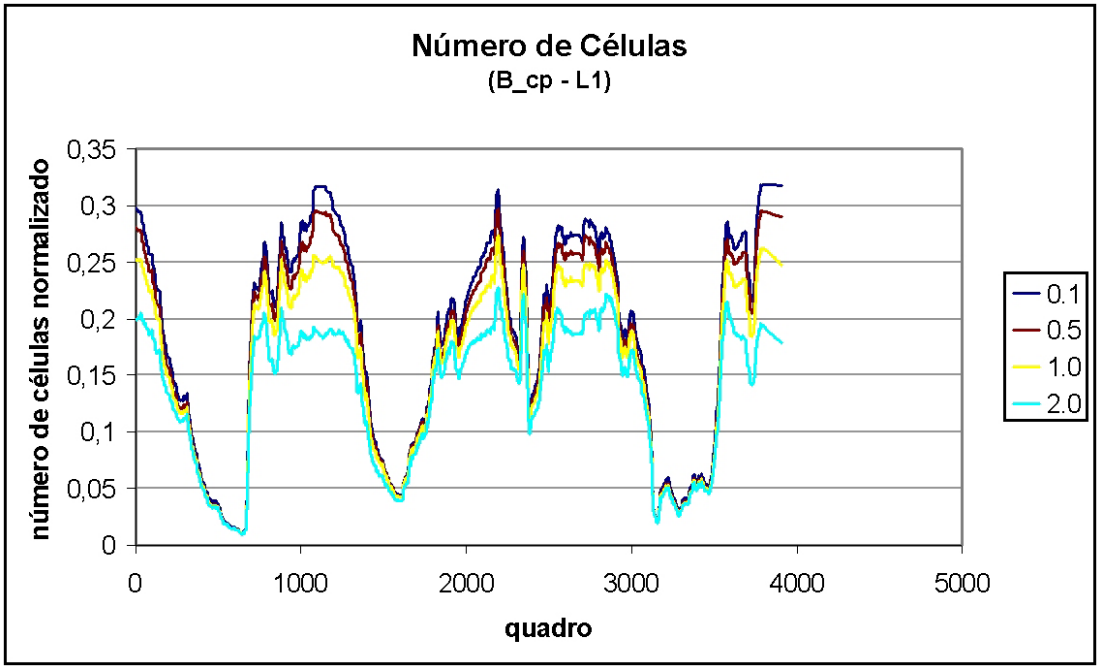


Gráfico 4 – Número de células desenhadas a cada quadro ao longo do caminho L1 na malha B_cp.

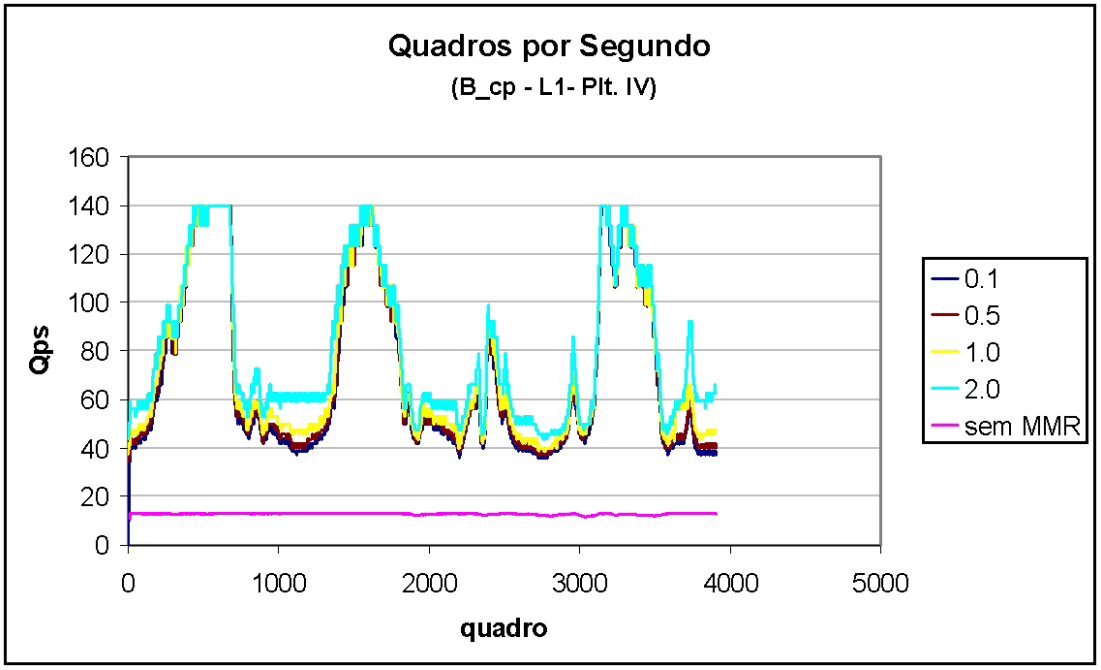


Gráfico 5 – Quadros por segundo ao longo do caminho L1 na malha B_cp. A média da curva "sem MMR" foi de 12,4 qps.

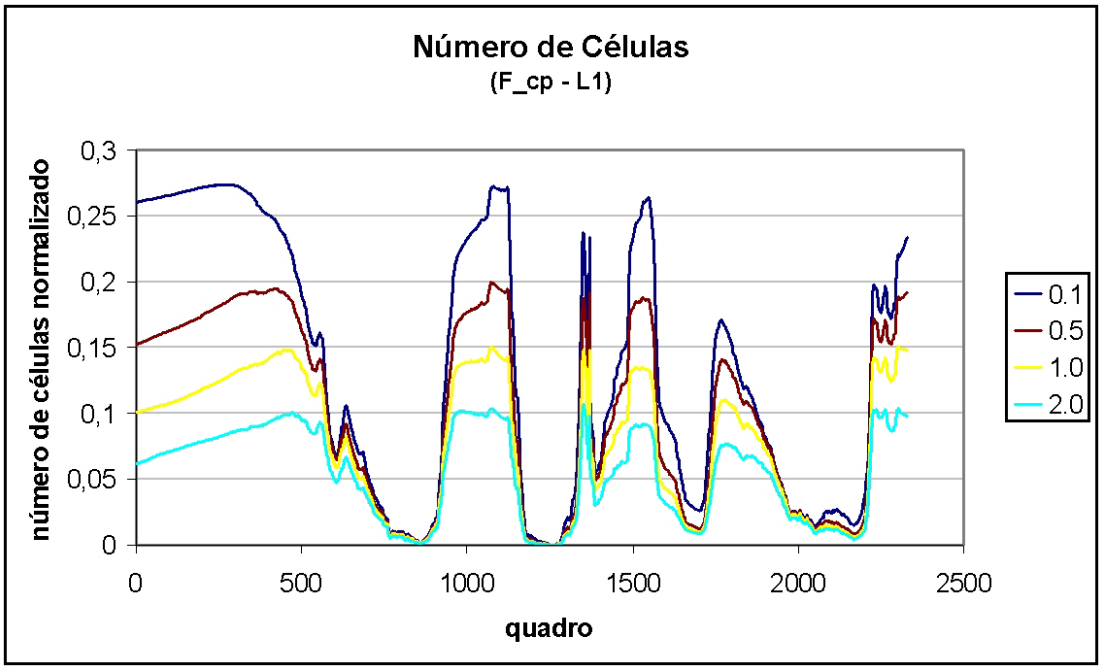


Gráfico 6 - Número de células desenhadas a cada quadro ao longo do caminho L1 na malha F_cp.

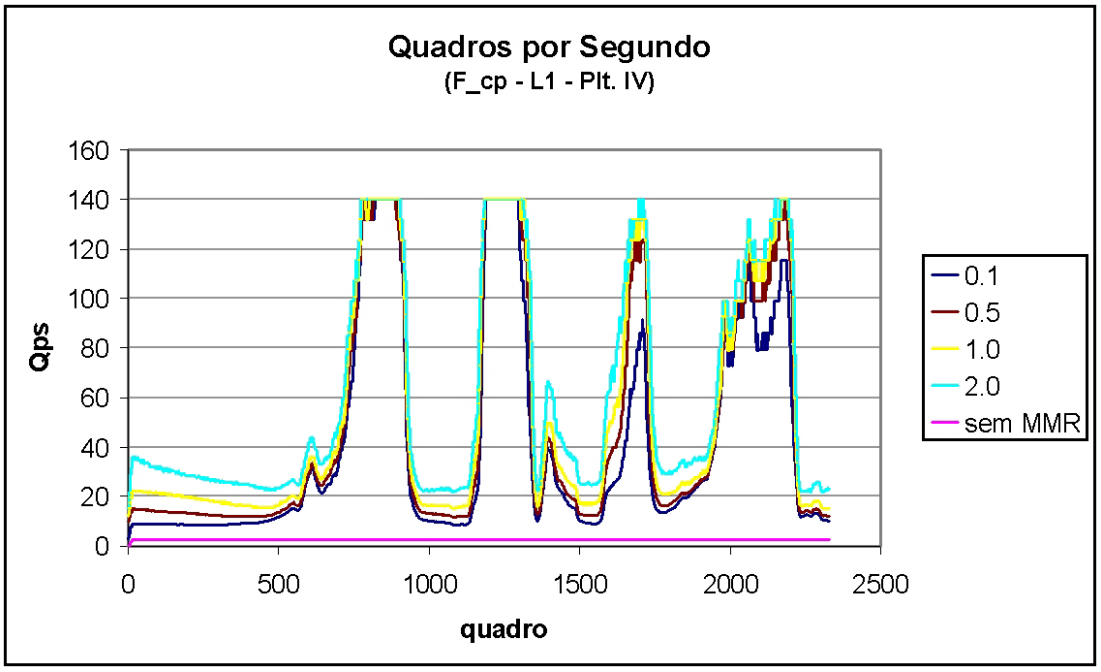


Gráfico 7 - Quadros por segundo ao longo do caminho L1 na malha F_cp. A média da curva "sem MMR" foi de 2,4 qps.

A Tabela 11 e a Tabela 12 apresentam as médias de quadros por segundo obtidas em cada curva das malhas B_cp e F_cp e também os piores casos de desempenho atingidos, oferecendo, assim, não só uma visão global do ganho de interatividade como também um limite inferior para esta.

Tabela 11 – Resultados, em quadros por segundo (qps), obtidos na malha B_cp sob a plataforma IV. Na primeira linha, a média de quadros por segundo ao longo do caminho e, na segunda, a menor taxa de interatividade atingida.

	Curvas				
	0.1	**0.5**	**1.0**	**2.0**	**sem MMR**
Média (qps)	57,77	**60,23**	**63,9**	72,48	12,83
Pior caso (qps)	33,85	**35,86**	**38,36**	43,64	10,7

Tabela 12 – Resultados, em quadros por segundo (qps), obtidos na malha F_cp sob a plataforma IV. Na primeira linha, a média de quadros por segundo ao longo do caminho e, na segunda, a menor taxa de interatividade atingida.

	Curvas				
	0.1	**0.5**	**1.0**	**2.0**	**sem MMR**
Média (qps)	17,13	**22,63**	**29,14**	40,18	2,4
Pior caso (qps)	7,58	**11,59**	**15,33**	21,68	1,91

Enquanto os ganhos de interatividade foram razoáveis sob a plataforma IV, os obtidos sob a plataforma III e ilustrados no Gráfico 8 e na Tabela 13, para a malha B_cp, foram menos expressivos, não conseguindo manter as taxas mínimas de interatividade (as taxas de interatividade começam a se tornar aceitáveis a partir de 6 qps [Tom99]).

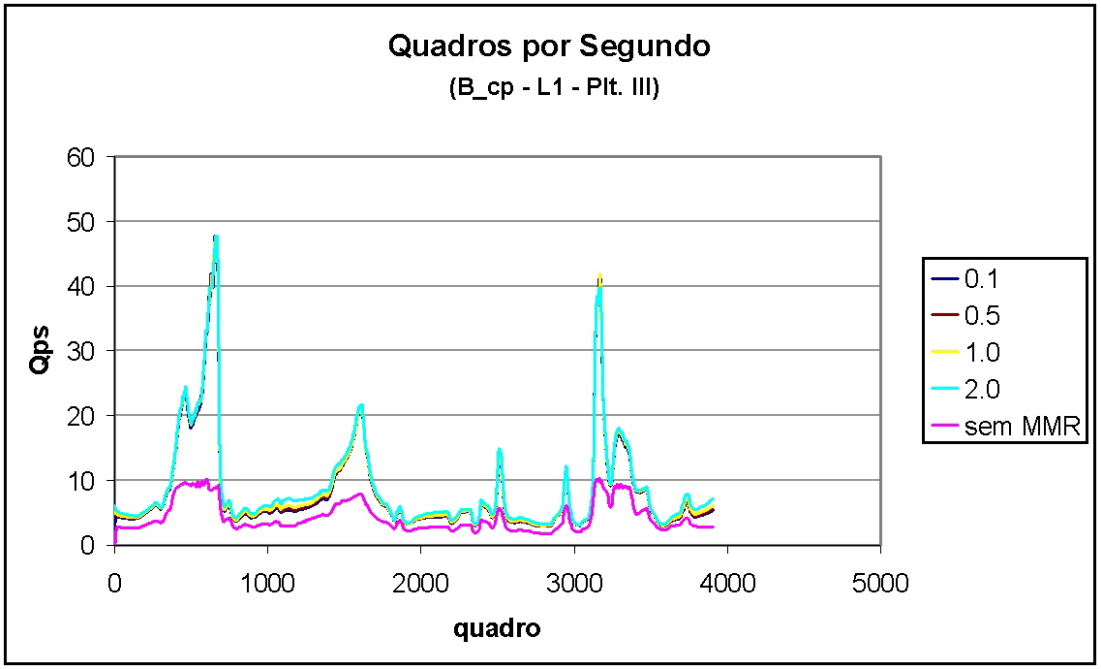


Gráfico 8 - Quadros por segundo ao longo do caminho L1 na malha F_cp sob a plataforma III. A média da curva "sem MMR" foi de 3,39 qps.

Tabela 13 – Relatório das taxas de interatividade (B_cp). Na primeira linha, as médias de quadros por segundo (qps) obtidas na malha B_cp sob a plataforma III e resolução de tela de 1000x800 *pixels*. Na segunda linha, os piores casos de interatividade em cada curva.

	Curvas				
	0.1	**0.5**	**1.0**	**2.0**	**sem MMR**
Média (qps)	5,47	**5,57**	**5,73**	6,15	3,39
Pior caso (qps)	2,84	**2,88**	**2,97**	3,09	1,69

O mau desempenho obtido pelo MMR na plataforma III se deve ao fato de que nem sempre, durante o percurso, o estágio de geometria foi o gargalo do canal de visualização[14]. Para analisar melhor essa questão, foi realizado um teste no qual o caminho L1 foi percorrido com a resolução de tela de 100x80 *pixels*, utilizando-se a malha B_cp sem o uso de MR. O objetivo desse teste é averiguar a influência do estágio de rastreio (*rasterization*) no canal de visualização, uma vez que ele é menos

[14] O canal de visualização é composto por três estágios conceituais: aplicação, geometria e rastreio [Tomas99].

exigido com essa resolução de tela. O Gráfico 9 compara os resultados obtidos com as resoluções de 1000x800 *pixels* e 100x80 *pixels*. A alteração significativa de desempenho permite concluir que, em algumas regiões da curva 1000x800, o estágio de rastreio foi o gargalo do canal de visualização.

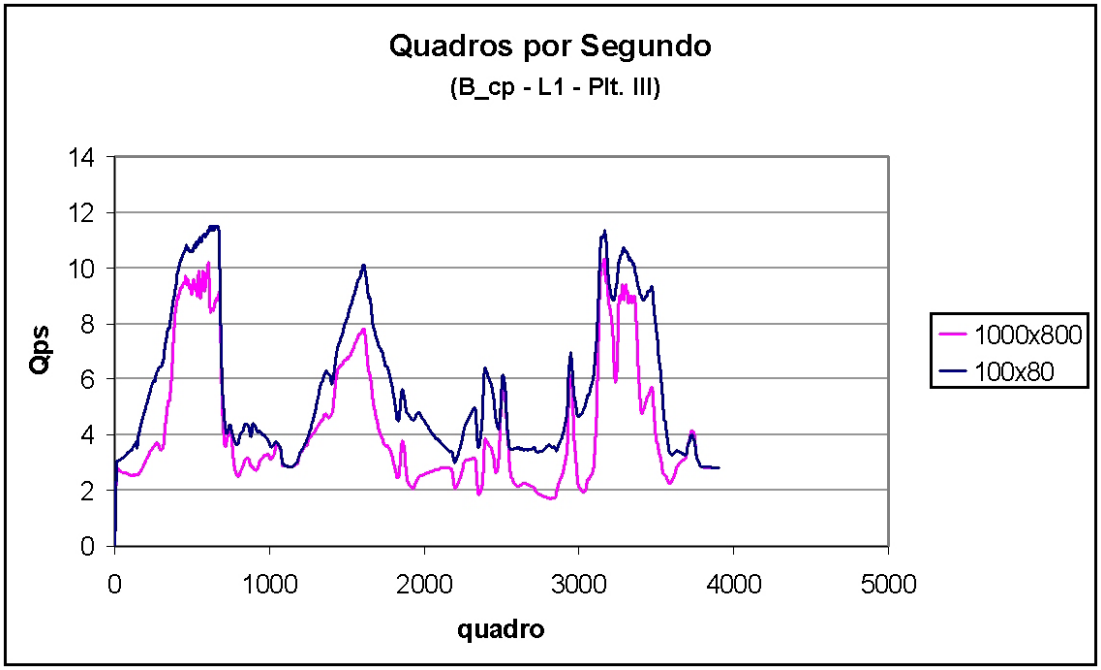


Gráfico 9 - Comparação entre as taxas de quadros por segundo obtidas sem o uso de MR e com diferentes resoluções de tela: 1000x800 e 100x80 *pixels*. A curva 1000x800 apresentou um média de 4,24 qps e a curva 100x80 de 5,7 qps.

Para finalizar a análise, os caminhos L1 foram percorridos com a resolução de tela de 100x80 *pixels* utilizando as malhas B_cp e F_cp. Objetivando aliviar apenas o estágio de rastreio, os parâmetros de erro projetado utilizados anteriormente foram diminuídos na mesma proporção da diminuição das dimensões de tela, ou seja, em dez vezes. Com isso, a quantidade de células enviadas para o canal de visualização mantém-se praticamente idêntica e os resultados podem ser comparados aos anteriores. As antigas curvas 0.1 são comparadas às novas curvas 0.02, por sua vez, as curvas 0.5 se referem às 0.1, as curvas 1.0 às 0.2 e, finalmente, as curvas 2.0 às 0.4.

O Gráfico 10, o Gráfico 11, a Tabela 14 e a Tabela 15 mostram os novos resultados obtidos. Pode-se constatar a melhora sensível do desempenho do MMR na malha

B_cp, na qual as taxas médias de interatividade passaram de aproximadamente 5,5 qps para mais de 15,0 qps. Na malha F_cp, as médias de interatividade das curvas foram maiores do que 5,0 qps, com exceção da curva 0.02, mas os piores casos foram aquém do necessário para permitir interatividade em tempo real.

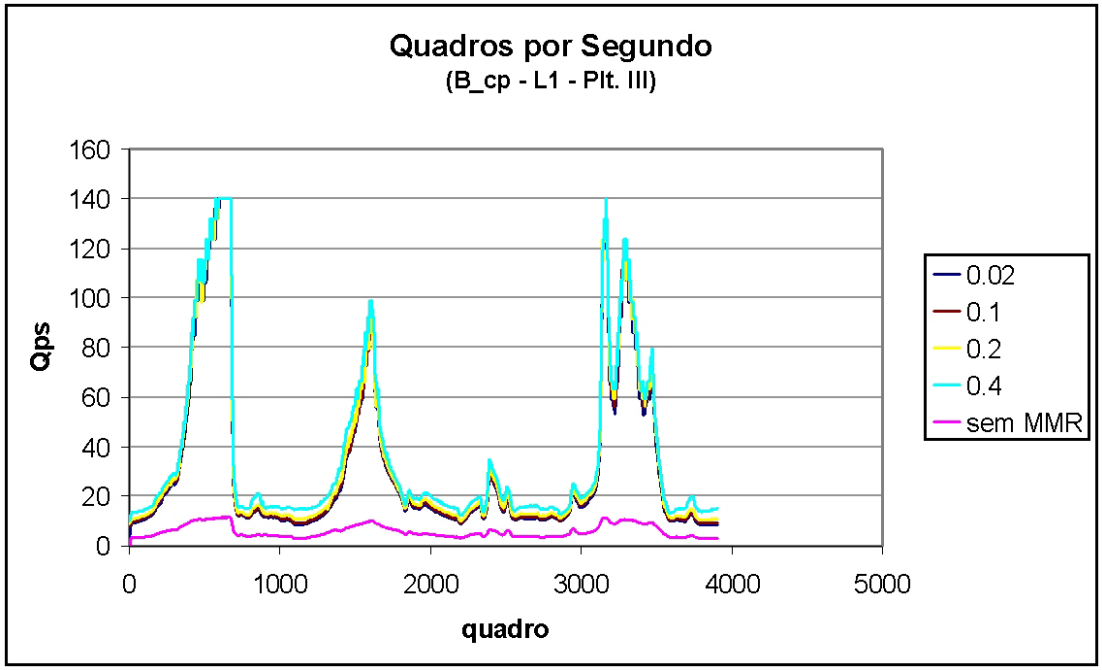


Gráfico 10 – Taxas de quadros por segundo obtidas no caminho L1 da malha B_cp sob a plataforma III e resolução de tela de 100x80. As curvas podem ser comparadas às dos gráficos anteriores.

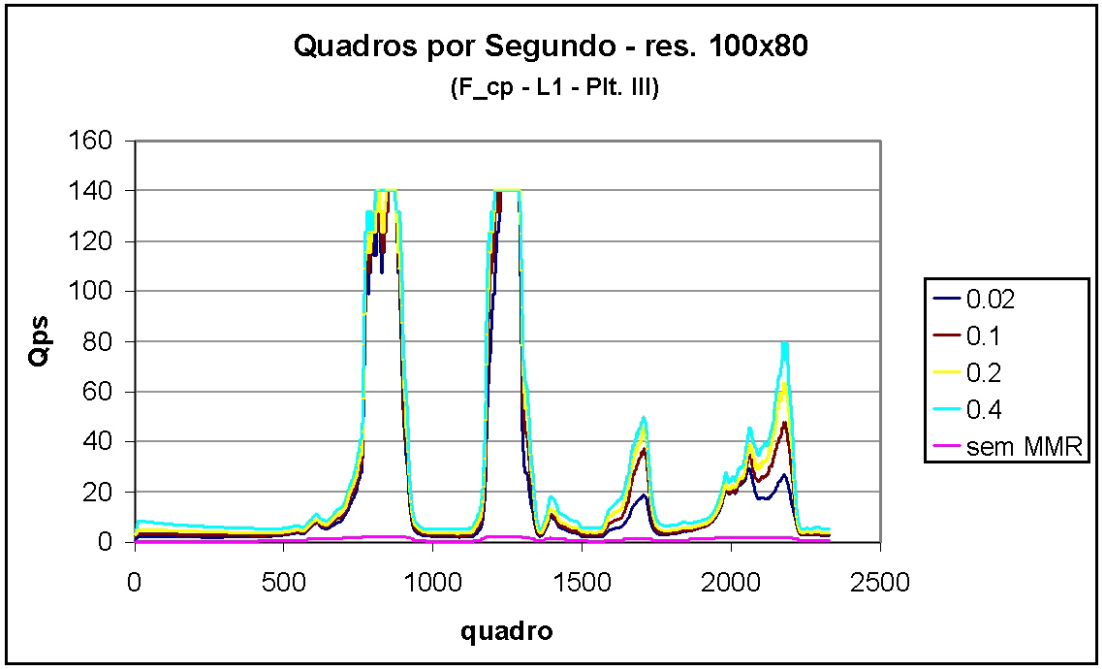


Gráfico 11 – Taxas de quadros por segundo no caminho L1 da malha F_cp utilizando a plataforma III e resolução de tela de 100x80.

Tabela 14 – Relatório das taxas de interatividade (plat. III). Na primeira linha, as médias das taxas de quadros por segundo (qps) obtidas na malha B_cp sob a plataforma III e resolução de 100x80. Na segunda linha, os piores casos de interatividade.

	Curvas				
	0.02	**0.1**	**0.2**	**0.4**	**sem MMR**
Média (qps)	16,18	**17,08**	**18,54**	22,15	4,78
Pior caso (qps)	7,43	**8,93**	**9,67**	12,32	2,32

Tabela 15 – Relatório das taxas de interatividade (F_cp). Tabela similar à anterior referente à malha F_cp sob a plataforma III e resolução de 100x80. Na primeira linha, as médias das taxas de quadros por segundo (qps) e, na segunda, os piores casos de interatividade.

	Curvas				
	0.02	**0.1**	**0.2**	**0.4**	**sem MMR**
Média (qps)	3,7	**5,03**	**6,63**	9,42	0,84
Pior caso (qps)	1,61	**2,5**	**3,34**	4,82	0,39

Conclusões Parciais

Os resultados obtidos demonstraram que, na plataforma IV, o MMR atingiu seu objetivo principal: o aumento considerável de interatividade, mantendo uma alta similaridade de aparência e possibilitando interatividade em tempo real. Entretanto, na plataforma III o MMR não obteve o mesmo êxito, embora ganhos tenham sido obtidos.

Os maus resultados obtidos na plataforma III devem-se (a) ao fato do gargalo do canal de visualização ter sido, muitas vezes, o estágio de rastreio e (b) à insuficiente redução de complexidade conseguida pelo MMR sem que a similaridade de aparência fosse penalizada.

O item (a) não é uma surpresa, pois é um fato conhecido que a técnica de MR atua diretamente sobre os estágios de aplicação e geometria, sobrecarregando o primeiro e aliviando o segundo. O ganho no estágio de rastreio é indireto e não se pode esperar que, sendo ele o gargalo, a utilização da técnica de MR obtenha ganhos de interatividade.

No caso da visualização de malhas de topologia 2-*manifold*, ou de outras malhas que representem superfícies em geral, o fato anterior passa, muitas vezes, desapercebido, pois a alteração da resolução da malha não afeta significativamente os cálculos do estágio de rastreio. A Figura 6.11 ilustra essa afirmação. Nela, percebe-se que os *pixels* a serem rastreados são exatamente os mesmos nas duas resoluções.

Entretanto, em malhas que representam um volume do espaço, como as malhas de simulação de reservatórios, a alteração da resolução altera a quantidade de *pixels* rastreados[15], como ilustrado na Figura 6.12. Assim, percebe-se que a implementação eficiente do estágio de rastreio é crítica nesse caso, pois o número de *pixels* rastreados pode crescer muito com o aumento da resolução.

---

[15] Está sendo considerado o tipo de visualização utilizado neste trabalho: visualização da fronteira das células das malhas. A afirmação não é válida para outro tipo de visualização, como, por exemplo, a visualização direta.

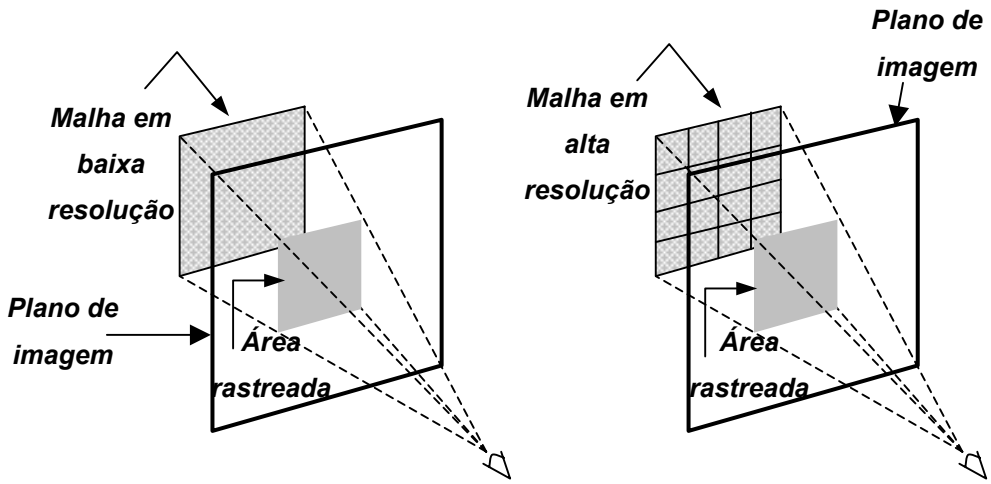


Figura 6.11 – Rastreio. As áreas rastreadas dos planos de imagem estão marcadas em cinza. Tanto em baixa resolução quanto em alta, a área é a mesma.

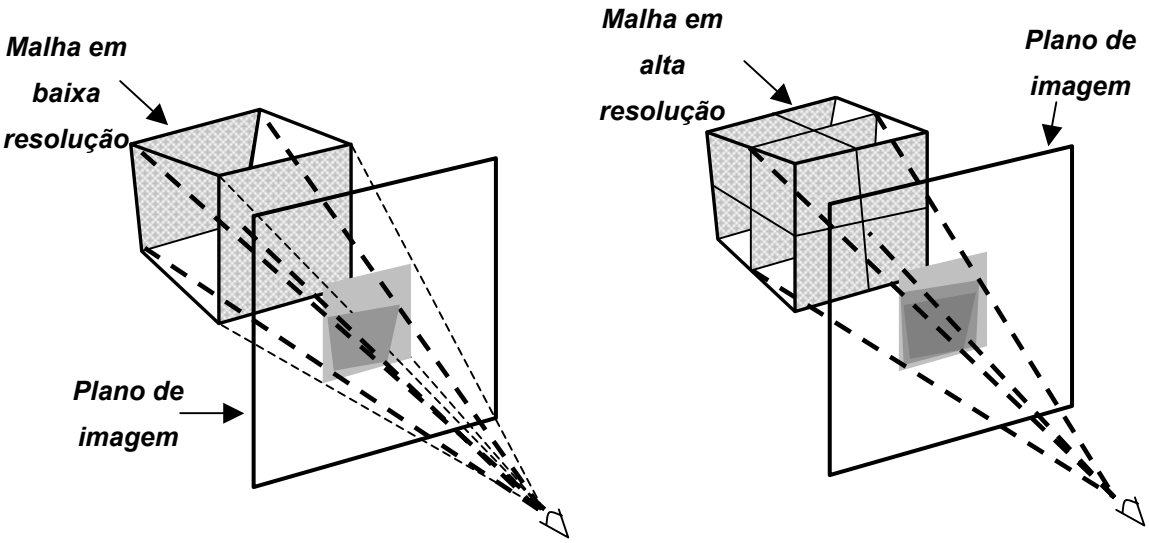


Figura 6.12 – Rastreio em volumes. As faces em destaque nas malhas são as mais significativas para o estágio de rastreio. No plano de imagem, estão as projeções das faces em destaque. As variações de cor nas projeções indicam quantas vezes os *pixels* estão sendo rastreados: as cores mais escuras indicam que os *pixels* foram percorridos mais vezes.

O item (b) é uma limitação do MMR desenvolvido e pode ser notado nos resultados obtidos na malha F_cp, através dos quais, mesmo com o estágio de rastreio aliviado,

percebe-se que a interatividade em tempo real não foi obtida, ou seja, o número de polígonos enviados para o canal de visualização ainda foi elevado demais.

Vale observar que, sob o aspecto do número de polígonos enviados para o canal de visualização, as malhas que representam volumes exigem, naturalmente, maior poder computacional do que as malhas que representam superfícies, pois, nas primeiras, o número absoluto de células que ocupam o volume de visão tende a ser maior (supondo malhas com a mesma densidade de células). Assim, voltando ao caso das malhas de simulação, é de se esperar que a sua visualização exija um alto poder computacional.

6.4 Características de Visualização

As características de visualização 5 e 6, que referem-se ao tempo gasto em pré-processamento e às taxas de interatividade, já foram analisadas nas seções anteriores. Cabe a esta seção verificar se as outras características de visualização foram atendidas adequadamente.

Para aumentar a confiabilidade dos resultados obtidos, a implementação do MMR aqui feita foi integrada ao aplicativo de visualização de malhas de simulação do fluxo de fluidos em reservatórios, o PosSim (Figura 6.13), um dos componentes do projeto GERESIM, o qual gerencia o processo global de simulação de reservatórios. Este aplicativo está sendo desenvolvido pelo Tecgraf (Grupo de Tecnologia em Computação Gráfica da PUC-Rio) em parceria com a Petrobras (Petróleo Brasileiro) e visa atender aos requisitos de visualização do problema considerado neste trabalho.

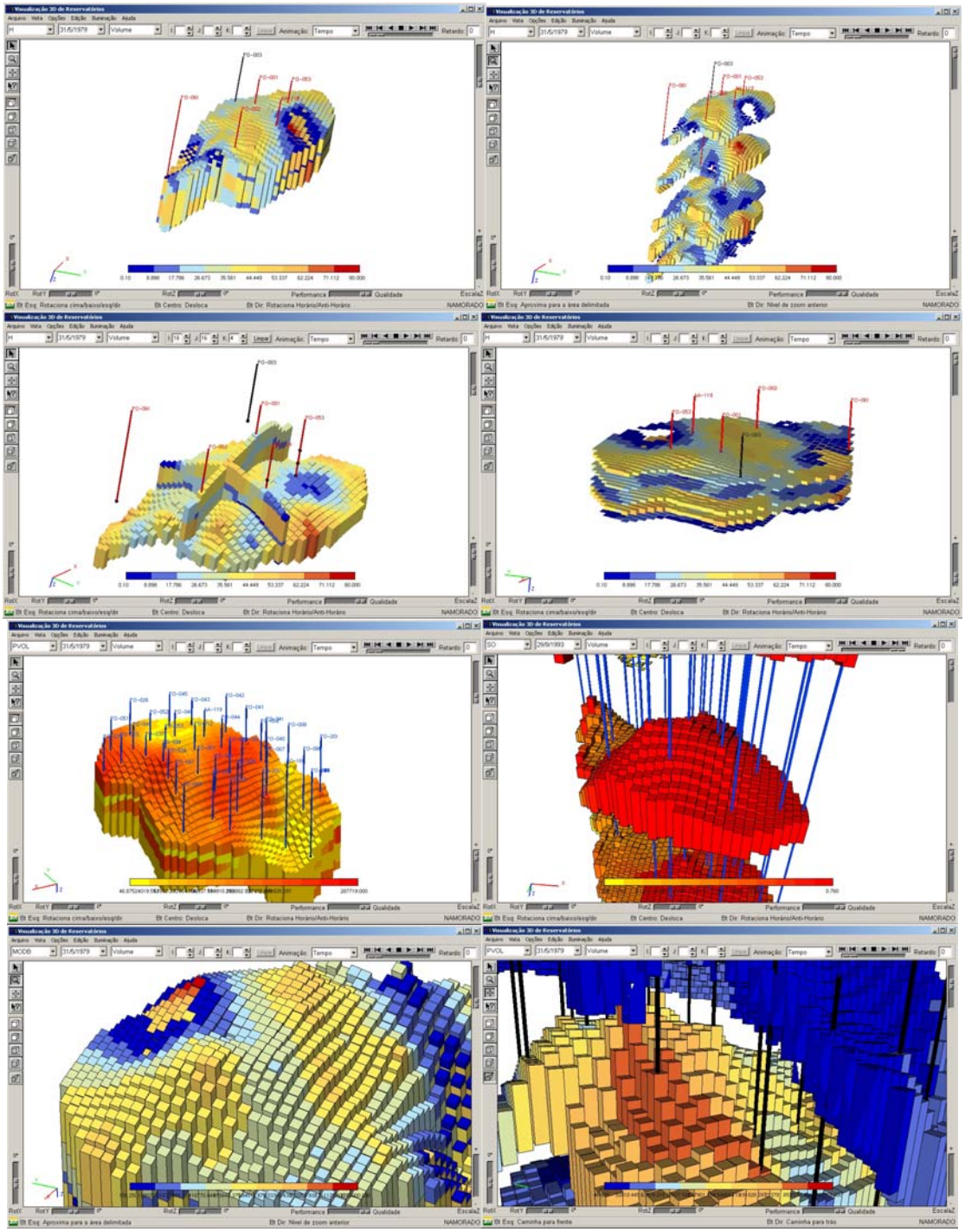


Figura 6.13 – PosSim (Visualizador 3D de reservatórios).

6.4.1 Propriedades Escalares

A inspeção das malhas de simulação em reservatórios é realizada através da associação à malha de uma propriedade escalar, a qual, normalmente, representa alguma grandeza física. Na visualização, isso é feito através de uma paleta de cores, que mapeia os valores escalares da propriedade a atributos de cores. Isso significa que a troca de uma propriedade física por outra está vinculada a uma possível troca dos atributos de cores das células da malha.

Visando gerar boas aproximações visuais, os algoritmos de extração de malhas consideram, também, o erro de propriedade em suas avaliações, tentando, com isso, adaptar a resolução da aproximação não só segundo a geometria, como também segundo as variações dos atributos de cores da malha.

Para atender às características de visualização 8 e 9, as quais requerem, no máximo, uma pequena perda de interatividade após a troca de uma propriedade ou durante sua animação temporal, o carregamento de uma propriedade na hierarquia de células hexaédricas é feito por demanda e consiste em percorrer a hierarquia, em tempo linear, calculando os valores e os erros de propriedade das macro-células através de cálculos já mencionados.

Uma vez que esses cálculos não alteram a estrutura da hierarquia, nada garante que a seqüência de colapsos que a gerou possua colapsos com baixos erros de propriedade. Ou seja, não há garantias de que o MMR seja capaz de gerar boas aproximações visuais.

Entretanto, a coerência espacial tipicamente existente em propriedades físicas aumenta a probabilidade dos colapsos efetuados durante a construção da estrutura de MR possuírem baixos erros de propriedade. Com base nisso, espera-se que o MMR seja capaz de representar suficientemente bem o reservatório na maioria das propriedades.

Esta seção apresenta os resultados obtidos com a estratégia de carregar as propriedades por demanda. Os testes realizados visaram analisar o quanto o erro

máximo de propriedade permitido interfere na capacidade do MMR de redução da complexidade das aproximações (Gráfico 12), desconsiderando o erro geométrico e a posição da câmera.

Os parâmetros utilizados para limitar o erro de propriedade permitido foram 0,05, 0,1, 0,15 e 0,2, números estes que estão normalizados pela diferença entre os valores máximo e mínimo da propriedade em toda a malha. Eles foram escolhidos em função da quantidade de intervalos existentes nas paletas de cores discretas geralmente utilizadas. Por exemplo, o parâmetro 0,05 é um bom valor a ser utilizado quando a paleta de cores possui 20 intervalos de cores, pois o erro visual de propriedade cometido não será maior do que um intervalo.

A análise foi feita na malha F_cp, escolhida por possuir um conjunto bastante significativo de propriedades, o qual representa uma vasta gama de grandezas físicas diferentes. Apenas parte dos resultados é apresentada no Gráfico 12, através do qual percebe-se que, na maioria das propriedades, o MMR consegue gerar malhas adaptadas às variações de propriedade, utilizando poucas células.

Em algumas propriedades (H, HT, RNTG e MLTZ), entretanto, o MMR precisou aumentar muito a resolução das malhas para obedecer ao limite máximo de erro permitido. É válido ressaltar que a propriedade MLTZ, que obteve os piores resultados, não é uma grandeza física: ela representa um fator de multiplicação associado a uma grandeza.

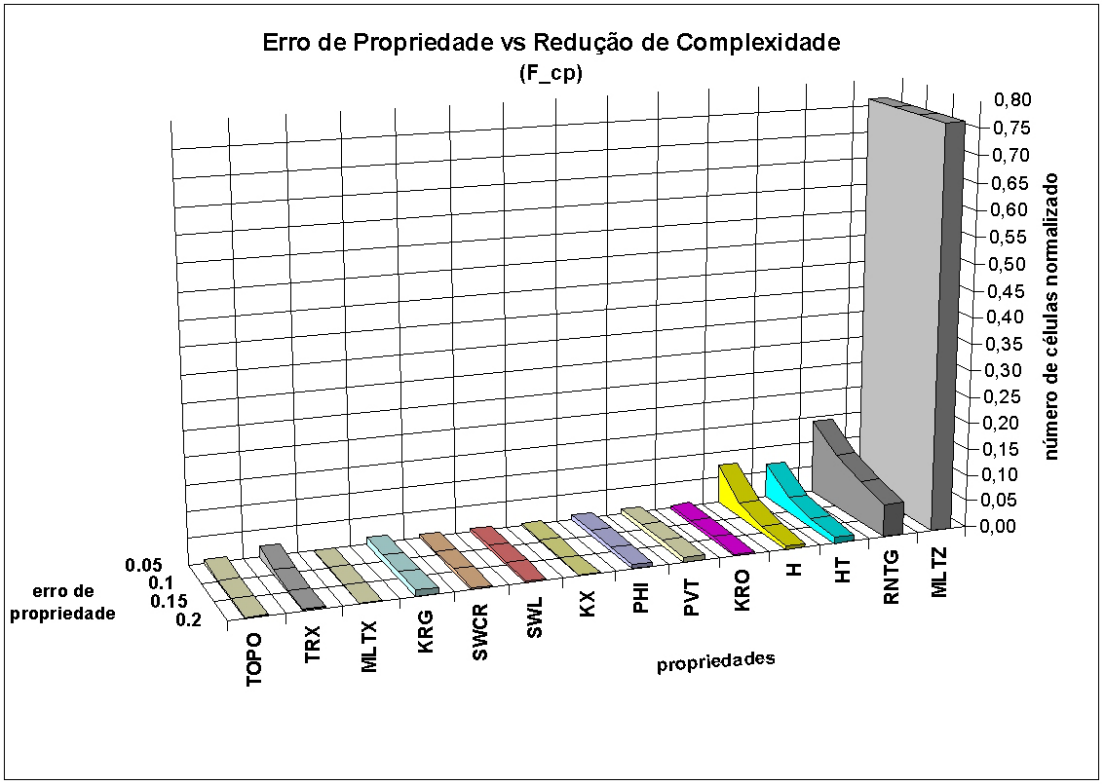


Gráfico 12 – Redução de complexidade nas aproximações em função do erro máximo de propriedade permitido. Os dados foram obtidos a partir do algoritmo de extração dependente do erro de propriedade. Os parâmetros de erro utilizados foram 0,05, 0,1, 0,15 e 0,2.

Conclusões Parciais

Confirmando as expectativas, os resultados mostraram que a capacidade de redução de complexidade do MMR, em geral, não é afetada pela consideração do erro de propriedade. Entretanto, a incapacidade de obtenção de altas taxas de redução, em algumas propriedades, pode impedir que a interatividade adequada seja alcançada sempre.

Com respeito a essa limitação, é válido ressaltar que, visando oferecer mais recursos ao PosSim, é possível modificar o algoritmo de construção da estrutura de MR, fazendo com que a operação de colapso considere o erro de propriedade em sua construção. O algoritmo modificado poderia, então, ser utilizado como uma opção ao usuário para visualizar as propriedades que obtiveram maus resultados com o primeiro algoritmo.

6.4.2 *Visualização da Grade (Wireframe)*

A característica de visualização 1 requer que o MMR seja capaz de desenhar o *wireframe* das células hexaédricas, de forma que uma grade de hexaedros seja visualizada.

A grade oferece ao usuário as informações de linhas, colunas e camadas, sendo interessante que a alteração da resolução da grade mantenha a capacidade de obtenção dessas informações. Por exemplo, a utilização de uma estrutura similar a uma *octree*, na qual os agrupamentos de oito células fossem baseados na vizinhança IJK da malha, responderia adequadamente ao problema de multi-resolução da grade, pois, a cada nível da árvore, teria-se uma nova grade, cujo intervalo entre linhas consecutivas representaria diferentes intervalos entre camadas, linhas e colunas na malha.

Entretanto, ao contrário dessa *octree*, a hierarquia de células hexaédricas desenvolvida neste trabalho não é capaz de gerar grades que atendam às necessidades de informação do usuário. A Figura 6.14 ilustra a visualização do *wireframe* das células através do MMR. Percebe-se que as linhas da grade de menor resolução, à direita, são interruptas, dificultando a obtenção da informação desejada.

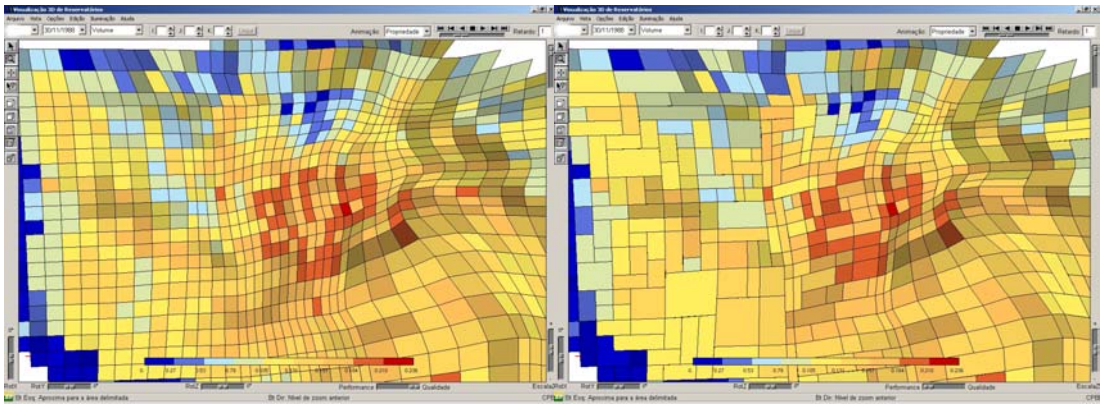


Figura 6.14 – Visualização da grade. Imagens geradas pelo MMR proposto nas quais nota-se a visualização do *wireframe* das células sob a vista superior, formando uma grade. À esquerda, a grade na maior resolução e, à direita, a grade em menor resolução.

Conclusões Parciais

A hierarquia de células hexaédricas é incapaz de gerar grades regulares porque a construção da hierarquia é feita por agrupamentos livres de células, sem a preocupação de obter grupos que, por nível da estrutura, sejam formados pelo mesmo número de células em I (ou em J ou em K), como acontece na *octree*.

É válido ressaltar que é a possibilidade de realizar agrupamentos livremente que permite à hierarquia de células gerar malhas bem adaptadas à geometria, ao contrário da *octree*. Esse fato foi comprovado, na prática, através da implementação da *octree* baseada no agrupamento de células vizinhas, oito a oito, que foi a primeira estratégia explorada neste trabalho. A Figura 6.15 compara os resultados visuais obtidos com a utilização das duas estruturas. As duas malhas possuem aproximadamente 21.700 polígonos.

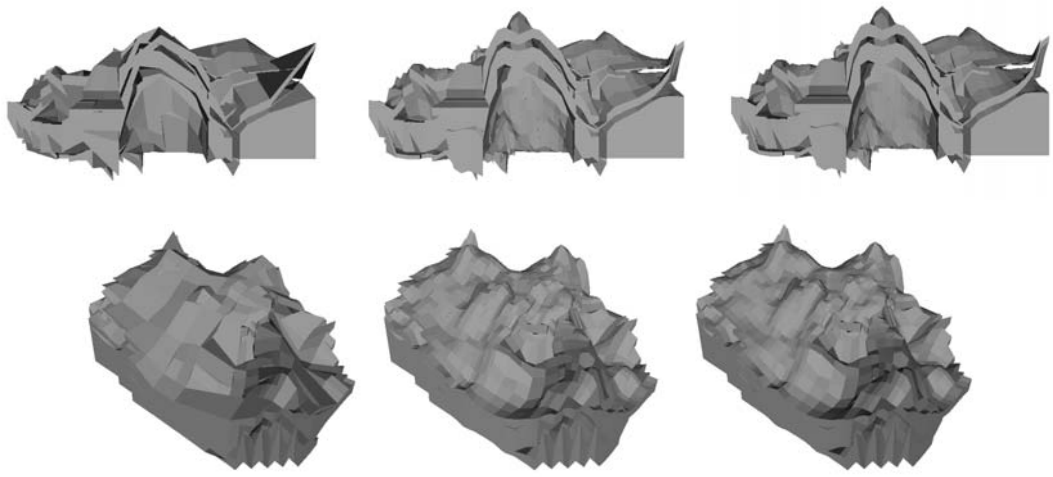


Figura 6.15 – Comparação visual entre os resultados obtidos com a estrutura *octree* e a hierarquia de células. As duas primeiras colunas contêm duas aproximações distintas da malha original, que é ilustrada na coluna da direita. Na coluna da esquerda, uma aproximação gerada pela estrutura *octree* e, na coluna do meio, uma aproximação gerada pela hierarquia de células proposta neste trabalho. Ambas aproximações possuem aproximadamente 21.700 polígonos (≈20% da malha original).

6.4.3 Separação entre Camadas e Escala em Z

As características de visualização 3 e 4 estão relacionadas a modificações na geometria da malha.

Particularmente, a característica 3 – a qual prevê uma possível escala em Z – é facilmente atendida pelo MMR aqui desenvolvido. A solução utilizada é bastante simples: no momento da renderização, multiplica-se o valor da escala que se quer realizar à geometria do reservatório, inclusive às *aabbs* associadas a cada nó, e ao erro geométrico armazenado pela estrutura de MR. Esse novo erro será uma majoração do erro que está sendo cometido com a escala em Z. O comportamento percebido é que, à medida que o usuário vai aumentando a escala em Z desejada, a resolução do reservatório tende a aumentar juntamente.

Uma solução alternativa, que não foi implementada, seria armazenar, na estrutura de MR, o vetor que representa o erro geométrico cometido, em vez da distância. Dessa forma, poderia-se multiplicar apenas a componente *z* do vetor para representar o erro após a efetuação de uma escala. Esta forma de variar o erro torna-o mais coerente com o erro que está sendo cometido com a escala em Z.

A característica 4, por sua vez, requer que as camadas do reservatório sejam visualizadas através de afastamentos, um para cada par de camadas vizinhas (Figura 1.3-c), fazendo com que a hierarquia de células só possa realizar colapsos entre células pertencentes à mesma camada.

Na prática, o MMR mostrou-se incapaz de reduzir a complexidade das aproximações quando as camadas são afastadas, impossibilitando o aumento das taxas de interatividade. Visando contornar esse problema, é utilizada, no PosSim, uma segunda hierarquia de células hexaédricas, obtida através de um algoritmo de construção modificado. A modificação realizada consiste em atribuir um custo infinito aos colapsos entre células de camadas distintas, fazendo com que eles sejam os últimos colapsos realizados durante a construção. Essa segunda hierarquia é capaz de reduzir com mais eficácia a complexidade das aproximações quando o afastamento entre camadas é efetuado.

Conclusões Parciais

Em relação à característica de visualização 3, a solução adotada obteve resultados satisfatórios, sendo que uma análise melhor se faz necessária apenas em relação à característica de visualização 4. Nesse caso, a incapacidade do MMR reduzir a complexidade das aproximações se deve ao fato de existirem poucos colapsos entre células de uma mesma camada na estrutura de MR. A causa disso é a super-amostragem existente nas colunas em profundidade, já citada em outras seções, fazendo com que, durante a construção da hierarquia, os primeiros colapsos a serem efetuados, em sua maioria, sejam justamente colapsos em Z, que agrupam células de camadas distintas.

A utilização de uma segunda estrutura para contornar esse problema acarreta o aumento do custo de construção da estrutura de MR e do consumo de memória. É importante observar que a segunda estrutura não gera aproximações tão boas quanto a primeira e, devido a isso, a primeira estrutura ainda é necessária. O Gráfico 13 ilustra o erro geométrico inserido nas aproximações, utilizando a segunda estrutura de MR. Ele é semelhante ao Gráfico 3, sendo obtido através do algoritmo de extração dependente do erro geométrico, mostrando valores de erro normalizados entre 0% e 20% da diagonal da caixa envolvente da malha. Comparando-o com o Gráfico 3, percebem-se resultados sensivelmente piores e, principalmente, a ausência de redução drástica das aproximações sem inserção de erro, indicando que realmente os colapsos em Z são importantes.

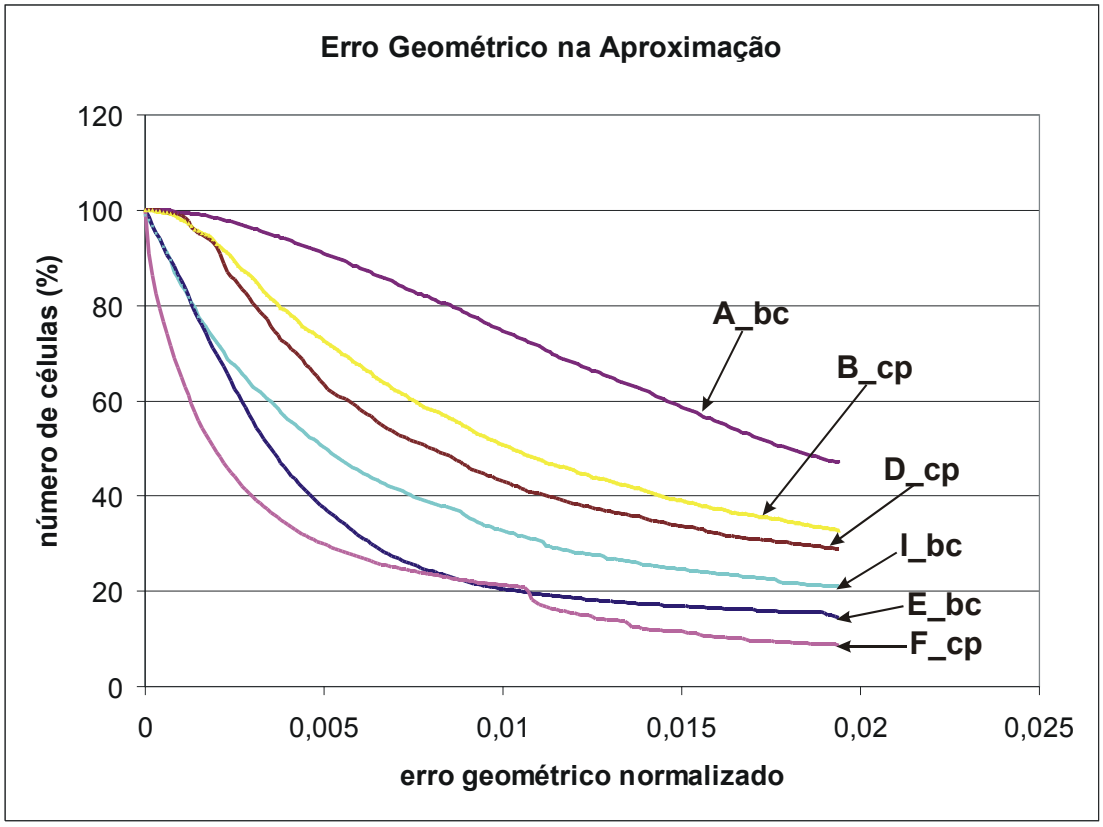


Gráfico 13 – Erro geométrico nas aproximações utilizando a segunda hierarquia de células hexáedricas.

Aqui, faz-se necessária uma observação complementar em relação à visualização de propriedades escalares (Seção 6.4.1): os resultados ruins obtidos na propriedade MLTZ também são causados pela necessidade de evitar colapsos entre células de camadas distintas, uma vez que MLTZ apresentou uma variação brusca de valores ao longo de Z. Dessa forma, a segunda estrutura de MR também pode ser uma solução para a visualização de propriedades.

7 Conclusões Gerais e Trabalhos Futuros

Este trabalho apresentou um modelo de multi-resolução para a visualização das malhas de simulação do fluxo de fluido em reservatórios naturais de petróleo. O modelo proposto é fortemente baseado nas características peculiares dessas malhas e nas características de visualização do problema em si, o qual é caracterizado como um problema de visualização científica.

Em síntese, o modelo de multi-resolução apresentado é um modelo hierárquico que mantém a célula hexaédrica como unidade básica de fragmentação e permite a extração de malhas adaptativas dependentes do erro geométrico, da câmera e do número desejado de polígonos.

Os resultados dos testes realizados permitiram obter conclusões específicas, que foram citadas anteriormente, dentre as quais destacam-se:

- algoritmo de construção da estrutura de MR utilizado permite uma boa relação entre custo de processamento e qualidade geométrica das aproximações;

- a extração das malhas pode ser feita em tempo real, utilizando-se equipamentos do tipo PC com aceleração gráfica, sendo que, para isso, preferencialmente, o estágio de rastreio da plataforma de *hardware* utilizada deve ser eficiente;

- as características de visualização foram atendidas parcialmente, destacando-se os bons resultados atingidos em relação aos requisitos de desempenho dos algoritmos; e

- o carregamento por demanda das propriedades escalares na estrutura de MR atingiu bons resultados, não prejudicando muito a capacidade de redução de complexidade das aproximações; entretanto, não há garantias de que bons resultados ocorram sempre.

7.1 Recomendações

A bateria de testes realizada permitiu estabelecer algumas recomendações importantes em relação aos requisitos de *hardware* necessários para um bom desempenho do MMR proposto.

A primeira recomendação direciona-se para a quantidade de memória disponível para o processo que executa o algoritmo de construção da hierarquia de células hexaédricas. Como dito na seção 6.2.1, o bom desempenho do algoritmo depende da inexistência de transferência de dados entre memória principal e auxiliar (*swap*). Sendo assim, os resultados da seção 6.2.2, que descrevem a quantidade de memória consumida pelo algoritmo de construção, devem ser utilizados como referência para a quantidade de memória que deve estar disponível para o processo. Os cálculos teóricos prevêem um consumo de 473×NC *bytes* de memória (onde NC é o número de células do modelo). Assim, para processar um modelo composto, por exemplo, por um milhão de células, seriam necessários 473 MB de memória disponível.

Outra recomendação é feita em relação ao *hardware* gráfico. Os testes de desempenho realizados na seção 6.3.3 mostraram um ganho de interatividade de, aproximadamente, 10 vezes em malhas de grande densidade de células. Esse valor pode ser considerado como base para a performance esperada e serve também para guiar a escolha do *hardware* gráfico a ser utilizado para a visualização de um determinado modelo. É válido ressaltar que essa recomendação é somente válida para o caso do gargalo do canal de visualização ser a geometria.

7.2 Trabalhos Futuros

O modelo de multi-resolução proposto pode sofrer algumas melhorias imediatas. Uma delas é definir um algoritmo de transição suave (*geomorphing*) entre duas malhas, ou seja, dadas duas malhas m_1 e $m_2 \in \mu'$, um parâmetro de transição $\alpha \in [0,1]$ e uma função de transição σ: $\mu' \times \mu' \times [0,1] \rightarrow \mu'$, achar $\sigma(m_1, m_2, \alpha) \in \mu'$, sendo que $\sigma(m_1, m_2, 0) = m_1$ e $\sigma(m_1, m_2, 1) = m_2$. Basicamente, o algoritmo de transição deve receber dois cortes na árvore, que representam os dois modelos de referência, e gerar o corte intermediário. Um algoritmo simplificado de transição já foi implementado e os resultados iniciais mostraram-se bastante motivadores. A transição implementada é aplicada somente a dois cortes que possuem um nível de diferença entre si na hierarquia. Assim, uma simples interpolação linear entre vértices (Figura 7.1) é aplicada para gerar a malha intermediária.

Outra melhoria é permitir a variação de cores dentro de uma macro-célula, através da interpolação das cores que estariam, agora, associadas aos vértices da macro-célula e não mais à macro-célula em si. Isso deve melhorar os resultados obtidos com o carregamento por demanda das propriedades escalares, resolvendo, por exemplo, o problema de propriedades que apresentam variações bruscas entre camadas consecutivas (Figura 7.2).

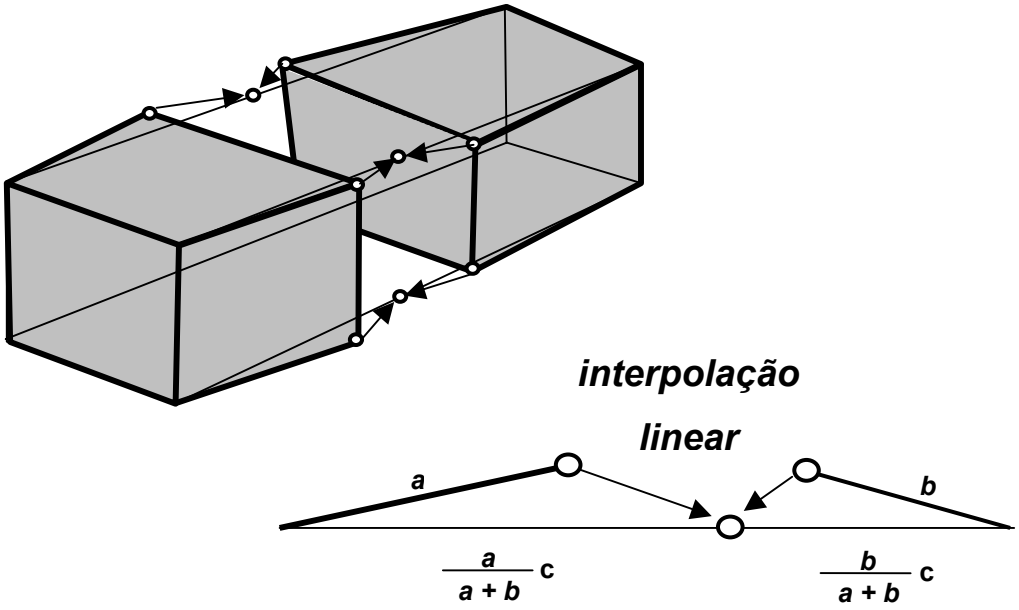


Figura 7.1 – Transição (*geomorphing*) por interpolação linear.

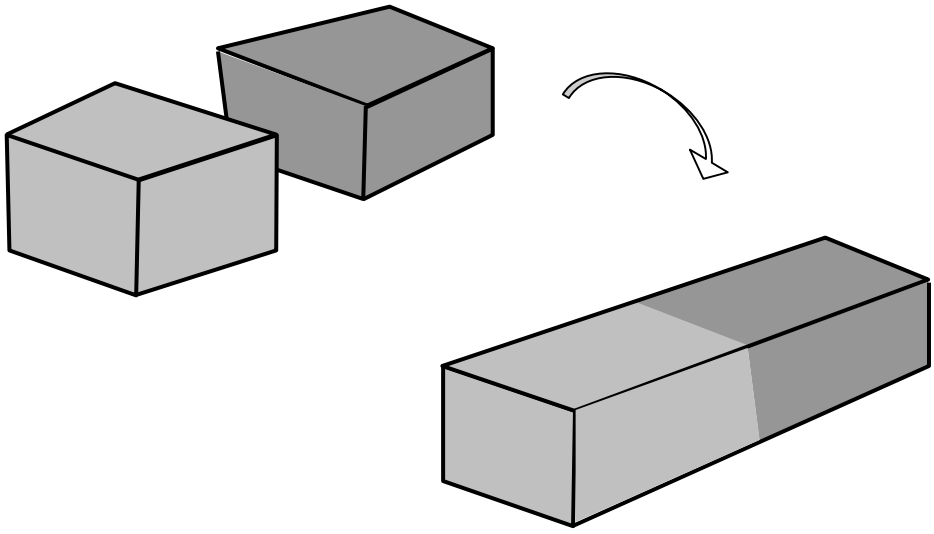


Figura 7.2 – Atributos de cores por vértices de célula. A variação das cores dentro da célula pode ser feita através da utilização de textura unidimensional.

Como uma terceira melhoria, pode-se pensar no aperfeiçoamento da operação de colapso utilizada pelo MMR, de forma que ela gere macro-células mais ajustadas à geometria do reservatório, principalmente em macro-células que representem baixas resoluções. Entretanto, deve-se analisar o impacto dessa nova operação no custo computacional do algoritmo de construção. Em particular, em contraposição à medida de erro utilizada neste trabalho, que, apesar de apresentar bons resultados, baseou-se numa estimativa intuitiva, pode-se utilizar uma medida que obedeça realmente a alguma norma conhecida ou que seja baseada em alguma medida estatística bem formalizada.

Além disso, pode-se estudar formas mais elaboradas de utilizar a hierarquia de células hexaédricas para visualização da grade. Por exemplo, a utilização de textura pode ajudar a resolver o problema.

Finalmente, está pendente um estudo aprofundado sobre a utilização da hierarquia de células em operações de busca, pois, apesar dessa estrutura não possuir a propriedade de altura logarítmica (*logarithmic height*), que é necessária para a realização eficiente desse tipo de operação [Pup97], a hierarquia de células apresentou um certo balanceamento nos testes realizados (a profundidade da árvore não ultrapassou o valor 45 nas malhas testadas). A operação de busca pode ser útil para responder, por exemplo, a perguntas do tipo: quais são as células que possuem valores de propriedade dentro do intervalo [i_o, i_f].

8 Referências Bibliográficas

[Aga94] P. K. Agarwal e S. Suri. “Surface approximations and geometric partitions”. Em *Proceedings of 5° ACM-SIAM Sympos. Discrete Algorithms*, páginas 24-33, 1994.

[Cig97] P. Cignoni, C. Montani, E. Puppo e R. Scopigno. “Multiresolution representation and visualization of volume data”. Artigo técnico C97-05, Instituto CNUCE – C.N.R., Pisa, Italia, Janeiro 1997.

[Cig98] P. Cignoni. “Scientific Visualization based on simplicial complexes”. Tese de doutorado, Universidade de Pisa, Pisa, Italia, Março 1998.

[Cor90] T. H. Cormen, C. E. Leiserson e R. L. Rivest. “Introduction to algorithms”. MIT Press e McGraw-Hill Book Company. 1990.

[Flo84] L. De Floriani, B. Falcidieno, C. Pienovi, and G. Nagy. “A hierarchical data structure for surface approximation”. Computers and Graphics, 8(2):475-484, 1984.

[Flo95] L. De Floriani and E. Puppo. “Hierarchical triangulation for multiresolution surface description”. *ACM Transactions on Graphics*, 14(4):363–411, October 1995.

[Flo97] L. De Floriani, E. Puppo, P. Magillo. “A formal approach to multiresolution hypersurface modeling”. Theory and Pratice of Geometric Modeling, Springer-Velrag. 1997.

[Gar98] M. Garland, A. Willmott, P. S. Heckbert. “Hierarchical face clustering on polygonal surfaces”. Artigo técnico.

[Gar99] M. Garland. “Quadric-based polygonal surface simplification”. Tese de doutorado, CMU-CS-99-105. Universidade de Carnegie Mellon, Pittsburgh, Estados Unidos. 9 Maio 1999.

[Ger98] A. Gerhardt. “Aspectos da Visualização Volumétrica de Dados Sísmicos”. Tese de Mestrado, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ, 1998.

[Her87] B. Von Herzen and A.H. Barr. “Accurate triangulations of deformed, intersecting surfaces”. *Computer Graphics (Siggraph 87 Proc.)*, 21(4):103–110, 1987.

[Heb95] D.J. Hebert and H.-J. Kim. “Image encoding with triangulation wavelets”. Em *Proceedings SPIE*, (2569(1)):381–392, 1995.

[Hof02] M. Hofmam, M. Gattass, W. C. Filho. “Tratamento Eficiente de Visibilidade através de Árvores de Volumes Envolventes”. Dissertação de Mestrado. Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil. Março de 2002.

[Hop96] H. Hoppe. “Progressive meshes”. *Computer Graphics*, Vol. 30 (SIGGRAPH ‘96). Páginas 99-108. 1996.

[Hop97] H. Hoppe. “View-dependent refinement of progressive meshes”. *Computer Graphics*, Vol. 31 (SIGGRAPH 97).

[Hop98] H. Hoppe, J. Poppovic. “Progressive simplicial complexes”. *Computer Graphics*, Vol. 30 (SIGGRAPH 96).

[IMEX00] Computer Modelling Group Ltd. “User’s Guide. IMEX.”.

[Lin94] T. Lindeberg. “Scale space theory in computer vision”. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1994.

[Low97] Low, Kok-Lim, T. S. Tan. “Model simplification using vertex clustering”. Em Symposium on Interactive 3D Graphics, 1997. ACM SIGGRAPH, páginas 75-82.

[Lue97] D. Luebke. “A survey of polygonal simplification algorithms”. Artigo técnico TR97-045, University of North Carolina- UCN-, Chapel Hill 1997.

[Math] http://mathworld.wolfram.com.

[Pup97] E. Puppo, R. Scopigno. “Simplification, LOD and multiresolution principles and applications”. *Eurographics ’97*, Vol. 16, 1997.

[Pre85] Franco P. Preparata and Michael I. Shamos. “Computational Geometry: an Introduction”. Springer-Verlag, New York, NY, 1985.

[Sca92] L. Scarlatos e T. Pavlidis. “Hierarquical triangulation using cartographics coherence”. CVGIP: Graphical Models and Image Processing, 54(2):147-161. Março 1992.

[Sch95] G. Schaufler e W. Stürzlinger. “Generating multiple levels of detail from polygonal geometry models”. Em Virtual Environments ’95 (Eurographics workshop), páginas 33-41. Janeiro de 1995.

[Tho01] J. E. Thomas *et al*. “Fundamentos de engenharia de petróleo”. Editora Interciência. Rio de Janeiro. 2001.

[Tom99] T. Möller, E. Haines. “Real-time rendering”.

[Wil83] L. Williams. “Pyramidal parametrics”. *Computer Graphics* (*SIGGRAPH ’83 Proc*), volume 17, páginas 1-11. Julho 1983.

[Vel00] L. Velho e J. Gomes. “Variable resolution 4-*k* meshes: concepts and applications”. Laboratório Visgraf, IMPA- Instituto de Matemática Pura e Aplicada. 2000.

[Vel01] L. Velho e D. Zorin. “4-8 Subdivision”. Laboratório Visgraf, IMPA- Instituto de Matemática Pura e Aplicada. 2001.

[Vel01b] L. Velho e J. Gomes. “Quasi 4-8 subdivision surfaces”. Laboratório Visgraf, IMPA- Instituto de Matemática Pura e Aplicada. 2001.

[Xia96] J. C. Xia, A. Varshney. “Dynamic view-dependent simplification for polygonal models”. Em *Proceedings of Visualization ’96*, páginas 327-334. Outubro de 1996.